

MAGAZINE
MSX
AÑO II
Núm. 15
Julio • Agosto
1986
300 Ptas.
1000
0110
0100
0111
1001
001
0000
El procesador de vídeo del SVI 318/328
TEST
Sony Hb 500P
BASIC
Diagramas de flujo
Código máquina, libros, programas
¿POR QUE ES LENTO EL BASIC?

**DIRECTOR:**
Juan Arencibia.
**COORDINADOR EDITORIAL:**
J. Ignacio Rey.
**COLABORADORES:**
Octavio López, Angel Zarazaga, Teresa Aranda, Ricardo García.
**DISEÑO:**
Benito Gil
Editada por:
**PUBLINFORMATICA, S.A.**
C/ Bravo Murillo, 377 - 5.° A
Tel.: 733 74 13
28020 Madrid.
Telex 48877 OPZXE
**PRESIDENTE:**
Fernando Bolín.
**DIRECTOR EDITORIAL REVISTAS DE USUARIOS:**
Juan Arencibia.
**DIRECTOR DE VENTAS:**
Antonio González.
**JEFE DE PRODUCCIÓN:**
Miguel Onieva.
**SERVICIO AL CLIENTE:**
Julia González.
Tel.: 733 79 69
**DIRECCION, REDACCION Y ADMINISTRACÍON:**
C/ Bravo Murillo, 377 - 5.° A.
Tel.: 733 74 13
28020 Madrid.
**PUBLICIDAD EN MADRID:**
Emilio García.
**PUBLICIDAD EN BARCELONA:**
Lidia Cendros.
C/ Pelayo, 12.
Tel.: (93) 301 47 00 Ext. 27-28.
08001 Barcelona.
Depósito Legal: M. 16.755-1985
Impreso en Héroes, S.A.
C/ Torrelara, 8. 28016 Madrid.
Distribuye:
S.G.E.L. Avda. Valdelaparra, s/n. Alcobendas (Madrid).
**DISTRIBUIDORES:**
VENEZUELA: SIPAM, S.A. Avda. República Dominicana, 541
ARGENTINA: DISTRIBUIDORA INTERCONTINENTAL BUENOS AIRES.

El P.V.P. para Ceuta, Melilla y Canarias, incluido servicio aéreo será de 300 ptas. sin I.V.A.

SUSCRIPCIONES:
Rogamos dirija toda la correspondencia relacionada con suscripciones a:
MSX
EDISA: Tel. 415 97 12
C/López de Hoyos, 141-5.°
28002 MADRID
(Para todos los pagos reseñar solamente MSX)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
MSX
C/Bravo Murillo, 377-5.° A
Tel. 733 74 13 28020 MADRID

Si deseas colaborar en MSX remite tus artículos o programas a Bravo Murillo 377, 5.° A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.




Las vacaciones son el momento ideal para dedicarse a todas aquellas actividades que, durante el resto del año, no se pueden realizar. Además, en estas fechas, en las que todo se paraliza o al menos se ralentiza, es agradable ver como MSX sigue su curso, un curso que cada vez es más firme.

La reciente feria, PROCESOS, que tuvo lugar en el Centro de Arte Reina Sofía, confirma el interesante futuro de estas máquinas. Ordenadores conectados a CD-ROM, a unidades de vídeo o a sintetizadores de sonido, han sido algunas de las interesantes opciones que hemos podido comprobar. Sin duda alguna, aún queda mucho que hacer y desarrollar, y nos ha sido grato comprobar como, día a día se van superando metas.

Cabe destacar que a la aparición del VG-8235, respondió Sony con el HB-500 P, que comentamos en estas páginas. Este equipo va a ser el enemigo principal, no sólo de otros ordenadores MSX de la segunda generación, sino de cualquier ordenador de 128K del mercado. Probablemente el lector se preguntará, cómo hemos dedicado tan poco espacio a dicho acontecimiento. Sencillamente hemos creído oportuno hacer una prueba, cara a cara, de estos dos equipos que publicaremos en meses sucesivos. No parece que exista un ganador de antemano, pero...

Por otro lado, el software empieza a ser novedad, principalmente debido a la llegada a nuestro país de programas de una importante empresa holandesa y de la aparición de programas para ordenadores de la 2.ª generación. Para los primeros veremos un simulador de vuelo en helicóptero que hará las delicias de cualquiera (ver sección de Software), mientras que para los segundos, pronto aparecerá un toolkit que facilitará la tarea de programación en estos potentes equipos.



# SUMARIO

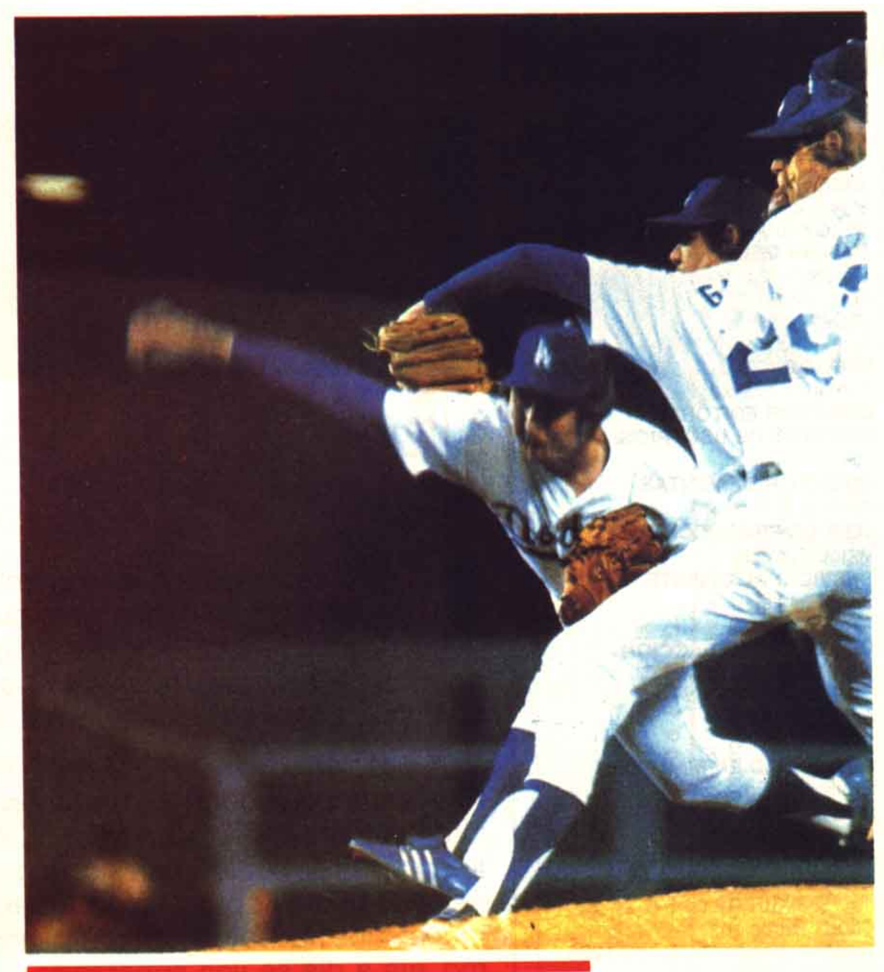
## 6

**Noticias:** Feria Procesos en el Centro de Arte Reina Sofía, algo muy especial. SERMA cambia de aires. Más libros de Anaya...

## 8

**¿Por qué es tan lento el BASIC?:** Aunque es el lenguaje de programación por excelencia de los ordenadores personales, cuenta con un gran defecto, su lentitud.



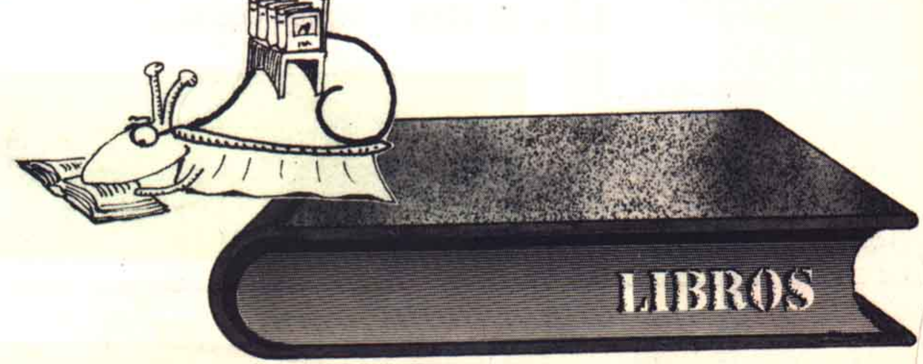


## 14

**Libros:** LOGO, de la tortuga a la Inteligencia Artificial. Un libro con grandes perspectivas que muestra una de las más importantes aplicaciones, la I.A. Por otro lado, MSX-El Manual Escolar, muestra al alumno el planteamiento y posterior solución de cualquier problema.

## 16

**Software:** Varios son los programas que se presentan este verano. Entre éstos tenemos, North Sea Helicopter (uno de los mejores simuladores de vuelo que hemos visto), Flight Deck, Beamrider, Gráficas de Gestión, Confused? y Drome.



NOTICIAS

# Más libros para el verano de Anaya

Proximamente aparecerán nuevos títulos de Anaya de contenido general, para todos los gustos y usuarios.

El «Diccionario de Informática» a un precio de 2.250 ptas. es una obra de consulta muy completa, que abarca desde los aspectos humanísticos de la informática hasta cuestiones puramente técnicas.

La «Programación del 8086/8088», donde se podrá estudiar el completo juego de instrucciones del procesador de Intel, es el libro idóneo para completar y adquirir un conocimiento de este microprocesador.

Para el programador o todos aquellos que necesiten de una pequeña introducción a los sistemas operativos va dedicado «Introducción a los Sistemas Operativos». Con unas 120 páginas y a un precio de 583 ptas, esta obra servirá como manual de introducción a otros sistemas operativos más conocidos como el UNIX, CP/M y MS-DOS.

También habrá un libro orientado al lenguaje C, «Programación en C con Tiny-C», es una obra desarrollada en base a una versión reducida del lenguaje C, que ha sido desarrollada para facilitar su aprendizaje. En 22 pág. y con un diskette incorporado, el usuario podrá encontrar una magnífica introducción a este lenguaje de programación. Sin embargo, el precio

# Idealogic y sus programas para Sony

Sony presentará en breve la nueva versión de lenguaje LOGO para MSX que ha elaborado IDEALOGIC S.A.

Además de poseer todas las instrucciones necesarias de cualquier lenguaje LOGO, incorpora a su vez, la posibilidad de amplair el lenguaje según las necesidades del usuario mediante la instrucción "USA".

Esta versión es similar sintácticamente a ACTI-LOGO, que esta empresa produce para ordenadores PC-MSDOS. Esto permite que listados de programas hechos en un PC-Compatible puedan ser introducidos directamente en el HIT BIT de Sony.

Por otro lado, IDEALOGIC, S.A. se va a encargar de la creación y desarrollo del software necesario para los ordenadores de la II generación, concretamente del Philips MSX-II. De estos cabe destacar la aparición del programa «Aerobic», con lecciones estructuradas según la necesidad de cada usuario, 1.250 pantallas animadas y con acompañamiento musical en cada uno de los movimientos.

(4.002 ptas. + I.V.A.) nos parece abusivo aunque lleve diskette.

No podían faltar los dedicados al usuario del IBM PC. En este caso, «dBASE III», está indicado para ser utilizado por dos tipos de lectores.

Primero, por el usuario de ordenador que quiere indagar sobre las características del dBASE III y, segundo, el usuario que ya tiene el programa y necesita una guía de referencia rápida y sencilla.

Para finalizar, tres obras orientadas al universitario. Estas son «Termodinánica y transmisión de calor», «Estadística» y «Métodos Matriciales». Con un precio que oscila entre 1.749 y 1.643, estos se presentan como una ayuda adicional, complementado los libros de textos. Cada uno trata un tema específico y concreto a nivel universitario.

## Un robot para jugar en casa

S.V.I. España presenta como gran novedad el primer robot para MSX. El brazo robot, consta de una base, brazo superior, antebrazo y muñeca.

Entre sus características más importantes, cabe destacar la posibilidad de utilizarlo tanto con el ordenador como con dos joysticks.

El Robotarm está destinado a los jóvenes con un conocimiento básico de programación y robótica elemental.

La facilidad de poder utilizarlo conectado al ordenador, le dan unas posibilidades fuera de lo común. Con un lenguaje de programación propio, ROGO (similar al LOGO) y sencillo, el usuario podrá ver y comprobar todas las cualidades y usos a los que se puede someter a un robot, éste se conecta al ordenador mediante el bus de expansión con un cartucho que encierra el juego de instrucciones.

En el programa de demostración, existe una rutina que comprueba el correcto funcionamiento de todas las piezas del robot.

También se puede utilizar sin el ordenador. Sin embargo, se necesitan dos joysticks, algo que no todos los usuarios poseen. De cualquier manera, con el ordenador se obtienen mejores resultados.

## Procesos, una feria muy particular

Del 26 de mayo al 20 de junio se celebró en el Centro de Arte Reina Sofía la feria Procesos, cultura y nuevas tecnologías.

En ella pudimos comprobar las diversas aplicaciones que, a pasos agigantados, están haciéndose muy conocidas y que están presentes en nuestra vida cotidiana.

La exposición se estructuró en tres ámbitos: memoria, comunicación y creación.

La holografía, proceso por el cual se realizan fotografías en tres dimensiones, fue la constante a lo largo de toda la feria. En cada rincón, había un motivo, un dibujo o una foto con esta peculiar técnica.





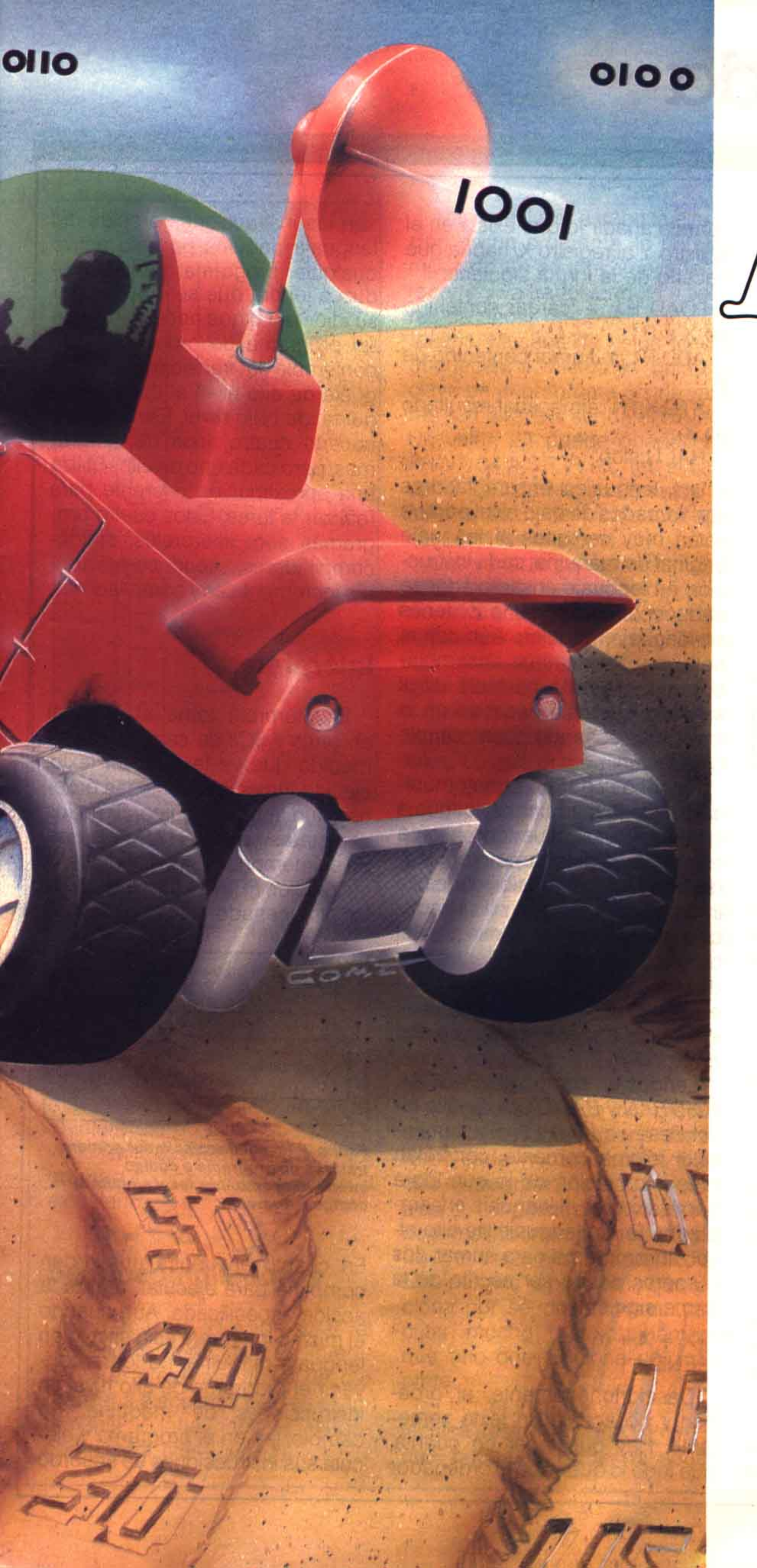
Los ordenadores también estuvieron presentes y MSX, no iba a ser menos. Sony, con su G-900, mostró unas cualidades que no existen en ninguna otra máquina. Similar al HB-500 P, este modelo con una unidad de superposición de vídeo se mostró superior a aparatos dedicados a tal fin, con la salvedad que éste cuesta alrededor de 300.000 ptas. mientras que una máquina de estas características ronda algunos millones. Con esta máquina se pueden realizar efectos especiales (tipo Guerra de las Galaxias) en casa.

Por otro lado, también se encontraba Yamaha, que con su modelo CX5M II mostró la increíble capacidad musical que posee.




# ¿Por qué es lento el BASIC?

**El lenguaje más popular de los ordenadores personales es, a su vez, el más lento. Sin embargo, su sencillez unida a su fácil aprendizaje y aplicación, lo han convertido en el más usado de los lenguajes de programación.**




Aunque es el lenguaje predominante entre los más conocidos, el BASIC tiene algunas desventajas muy evidentes cuando se le compara con otros lenguajes de programación, tales como *PASCAL* o *FORTRAN*, dos lenguajes de alto nivel dedicados a aplicaciones específicas. El *PASCAL*, está más generalizado que el *FORTRAN* (cuyas siglas significan FORmula TRANslator-traductor de fórmulas), empleando fundamentalmente en cálculos científicos y matemáticos, aunque no está implementado en los ordenadores personales. La primera pega que podemos poner al *BASIC* frente a estos lenguajes es su lentitud. Esto es fácilmente demostrable, sólo basta con crear cualquier figura animada y desplazarla por la pantalla.

¿Por qué motivo los programas desarrollados en *BASIC* se ejecutan más despacio que otros escritos en lenguajes de alto nivel o ensambladores?

Para ver en que consiste el problema, veamos cómo se ejecutan los programas en un ordenador. Todos los ordenadores, desde el personal hasta el *mainframe*, ejecutan los programas llevando a cabo instrucciones en lenguaje máquina. Estas instrucciones son órdenes codificadas eléctricamente, como ceros (0) y unos (1) que realizan operaciones extremadamente sencillas, como por ejemplo, «leer un dato de memoria y almacenarlo en un registro». Este grupo de operaciones se denominan juego de instrucciones y ya vienen incorporadas al ordenador en la memoria *ROM*.

Un programa de ordenador consiste en una larga serie de estas instrucciones elementales. Cuando la computadora está ejecutando un programa, busca es-

tas instrucciones en la memoria y las va ejecutando. Cada una de estas cumple una pequeña función, sin embargo, el conjunto de estas hacen un programa complejo.

Las instrucciones de lenguaje máquina se almacenan en memoria en grupos de *bits*. Los *bits* pueden ser representados en el sistema binario como cadenas de unos y ceros (ver «Sistemas de numeración», MSX núm. 2)

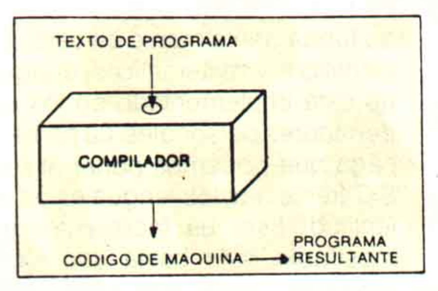



***Figura 1: Un intérprete lee el texto del programa y lo lleva a instrucciones de programa.***

Sin embargo, esto ha sido siempre un problema. Con los primeros ordenadores había que tener en cuenta qué combinación de ceros y unos pertenecían a qué instrucción. Por este motivo, los programadores, cansados de preocuparse sobre este gran inconveniente, han creado sistemas de codificación para grabar instrucciones. Estos sistemas de codificación son los llamados lenguajes ensambladores. Una orden, para añadir un número en el lugar X de la memoria, se puede escribir de la forma siguiente:

ADD R1, X

Los programas que traducen estas simples nociones a equivalentes de instrucciones en lenguaje máquina son llamados ensambladores.

Por ejemplo, si se quiere realizar un programa ensamblador que permita añadir lo que exista en el registro Y al registro X, habría que hacerlo de la forma siguiente:

LOAD R1, X ;cargar contenido de X a R1.

ADD R1, Y ;sumar contenido de Y.

STO X, R1 ;almacenar resultado en X.

HALT ;stop.

Los lenguajes ensambladores son llamados de bajo nivel porque están muy cercanos al lenguaje original de máquina; cada instrucción en lenguaje ensamblador se traduce directamente a órdenes en lenguaje máquina. Aún con la notación de lenguaje ensamblador, sigue siendo bastante difícil saber lo que está pasando en la máquina echando una simple ojeada.

Entonces ¿por qué no comunicarse con la máquina en lenguaje corriente? Dado que todo idioma está lleno de complejas irregularidades y múltiples posibilidades de interpretación, es muy difícil traducir a un lenguaje accesible para el ordenador. Pero los ordenadores dieron con un compromiso entre el dioma corriente y el lenguaje ensamblador: el lenguaje de alto nivel.

Lenguaje como el *PASCAL*, *FORTRAN* y *BASIC* son de alto nivel. Pese a que no son exactamente el idioma corriente, concretan más información de lo que logra habitualmente el lenguaje ensamblador. En un lenguaje de alto nivel, un programa para sumar dos números puede ser escrito de la forma siguiente:

X = X + Y.
END

Desafortunadamente, el ordenador no acepta un texto semejante. Hay que tener en cuenta, que todo lo que sabe el ordenador son las instrucciones propias del lenguaje máquina, por este motivo cualquier programa que se introduzca tendrá que ser traducido a su idioma. Lo que necesitamos es un programa de ordenador que contemple la transición del lenguaje de alto nivel al código máquina, de bajo nivel. Esto pueden hacerlo cuatro tipos de programas, pero cada uno de ellos utiliza una aproximación diferente para realizar la tarea. Estos cuatro programas son: intérpretes, seudocompiladores, seudocompiladores incrementales y compiladores.

## Intérpretes

El intérprete toma el material en bruto *ASCII* de caracteres y, a medida que va leyendo el texto, ejecuta inmediatamente las órdenes que le da el código. Si en los caracteres se lee *PRINT* X, busca en su memoria el significado de la instrucción *PRINT* e imprime el valor almacenado en la posición X.

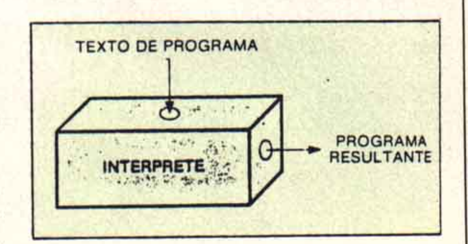



***Figura 2: Un seudocompilador primero traslada un programa a código intermedio; entonces lee y ejecuta el código intermedio.***

En cuanto encuentra una orden completa para ejecutar, realiza la acción especificada. Aún cuando el mismo intérprete está escrito en lenguaje máquina, no logra convertir el programa de alto nivel en instrucciones de máquina. En cambio, lee en el programa y ejecuta sus instrucciones de acuerdo

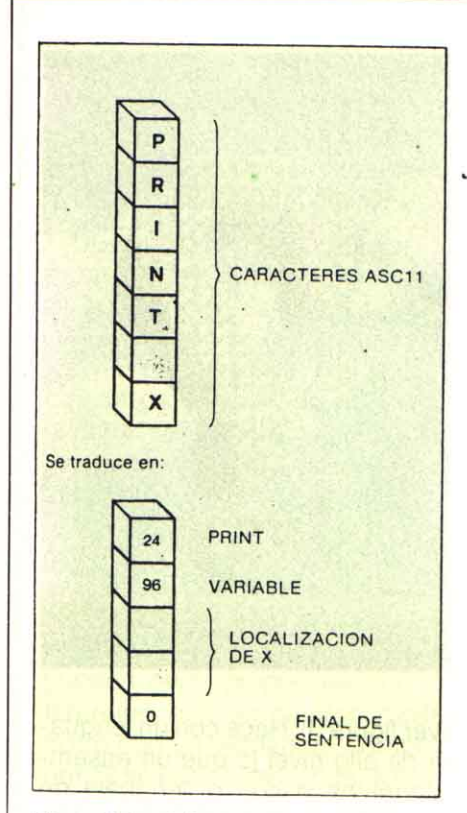



***Figura 3:* Los tokens sustituyen a las palabras claves y a las variables.**

al vocabulario de palabras clave existentes en el intérprete. Con esta aproximación, sin embargo, el intérprete debe perder tiempo en leer y traducir el texto. En consecuencia, los intérpretes son lentos. Por otra parte, al traducir programas según se va introduciendo, el intérprete no tiene que perder varios minutos traduciendo el programa a código máquina antes de ejecutarlo.

## Seudocompiladores

Una manera de apresurar el proceso de traducción es «predigerir» el programa antes de comenzar a ejecutarlo. Esto es exactamente lo que hace el seudocompilador (ver figura 2).

Antes de llevar a cabo la ejecución de un programa, la primera mitad de un seudocompilador lo recorre y va traduciéndolo a valores de uno o dos *bytes* llamados *tokens.* Muchos de estos *tokens* obedecen a diferentes palabras clave en el programa (estos valores son distintos según el *BASIC*). Un valor, 24 puede obedecer a *PRINT,* 34 a *STOP,* etc. (en el caso del *BASIC* de Microsoft, todos los *tokens,* sean del ordenador que sea, son los mismos).

Una variable puede ser almacenada como *token,* seguida de un número entero que indique la localización en la memoria. Todos los espacios, retorno de carro y comentarios en el programa se suprimen porque el ordenador no tiene ningún tipo de relación con ellos. Lo que originariamente era: (ver figura 3).

Los valores *tokens* que comprenden las sentencias resumidas del programa son llamados seudocódigos o códigos intermedios. El lenguaje de programación *PASCAL* se maneja habitualmente con seudocompilador.

Una vez traducido el programa a seudocódigo, queda listo y ejecutado por la segunda mitad del seudocompilador, el intérprete de seudocódigo.

A diferencia de un intérprete, el seudocompilador no tiene que traducir cada vez una orden. Dado que el programa ha sido reducido a series de *tokens* de 1 *byte,* el seudocompilador simplemente lee en un *byte* y va a ejecutar esa acción. Esto permite al programa que funciona por seudocompilador, ejecutar mucho más rápidamente que uno que funciona por intérprete.

El mayor inconveniente de esta técnica reside en que, antes de que pueda funcionar el seudo-



***El BASIC, el más popular de los lenguajes de programación es, a su vez, el más lento.***

compilador, debe parar y traducir el programa a un código intermedio. Es un proceso rápido: la mayor parte de los programas pequeños tardan entre 10 y 15 segundos en completar el procedimiento. Sin embargo, este produce un lapso significativo entre el momento en que se le dice al ordenador que ejecute el programa y el momento en que comienza a hacerlo (algunos sistemas permi-

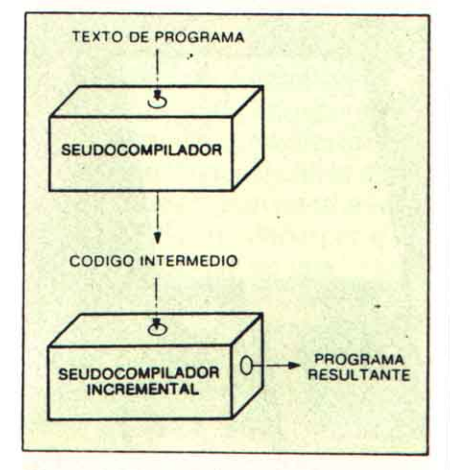
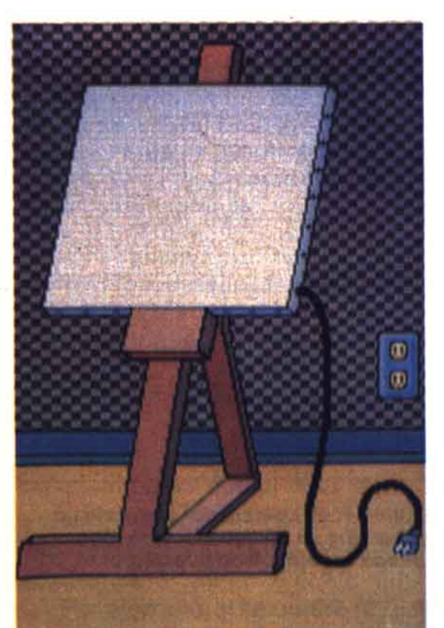



***Figura 4:* Un compilador convierte un programa en código máquina. Este puede ser ejecutado directamente.**

ten almacenar el código de inmediato para uso repetido, pero también en ese caso hay que esperar por la retraducción cada vez que se produce una modificación en el programa). Además, aún cuando traducir el código inmediato es más rápido que traducir texto desecho, sigue siendo un proceso demasiado lento dado que los programas seudocompilados son unas diez veces más rápidos que los traducidos.

## Seudocompiladores incrementales

Es aburrido tener que esperar aunque sólo sean unos pocos segundos para que seudocompile un programa. Para solucionarlo, los diseñadores de lenguajes dieron con el seudocompilador incremental *(IPC)*. Con un *IPC*, el editor de texto en que escriba su programa, está fusionado con el seudocompilador. A medida que teclea un programa, el *IPC* sigue lo que se está tecleando. Y al presionar el retorno del carro *(CR)*, el *IPC* inmediatamente traduce a seudocódigo la línea que se acaba de redactar y la almacena. Porque sólo traduce una línea por vez, el proceso no lleva más de unas milésimas de segundo. Es tan rápido que ni se percibe que algo haya pasado. El *IPC* puede o no mantener una copia de la línea en su redacción original.



Dado que la mayor parte de la seudocompilación ya ha sido llevada a cabo antes de que el programa se ejecute, un *IPC* puede comenzar con la ejecución del programa casi cuando se teclea *RUN*. El *IPC* puede realizar algunas tareas antes de comenzar, pero no provoca ninguna demora evidente.

Al igual que un seudocompilador normal, un *IPC* va mucho más rápido que un intérprete porque el volumen de la labor de traducción se completa mucho antes de que el programa comience a ser ejecutado. Pero el *IPC* tiende a funcionar con lentitud significativamente mayor que los seudocompiladores no incrementales, porque la mayor parte está diseñada para producir un código intermedio compacto.

Los sistemas *BASIC* son generalmente escritos como *IPC*. El intérprete *BASIC* de Microsoft es realmente un *IPC*. De hecho, la mayoría de los sistemas a los que se hace referencia como intérpretes son en la actualidad *IPC*. Los verdaderos intérpretes son tan lentos que muy pocas veces se emplean.

## Compiladores

Un verdadero compilador toma el texto inicial *ASCII* de su programa y lo traduce a código máquina, y no sólo a un código intermedio

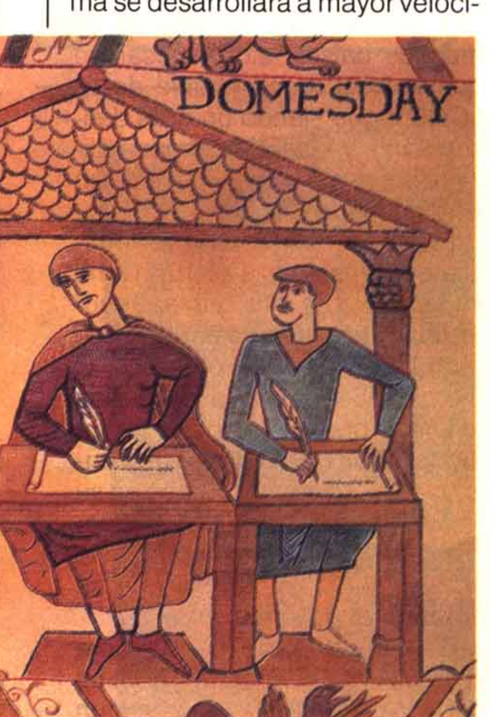


(ver figura 3). Hace con un lenguaje de alto nivel lo que un ensamblador hace con el lenguaje de ensamblaje. Cuando se le dan instrucciones a un ordenador para un programa que ha sido compilado, debe saltar al comienzo del



código máquina y empezar a ejecutar. Porque no es necesaria ninguna traducción, el programa puede transcurrir con mucha velocidad, hasta 500 veces más rápido que un programa interpretado.

Desgraciadamente, traducir un programa de lenguaje de alto nivel a código máquina es complicado y muy lento. Al ejecutar cientos de instrucciones por segundos, el ordenador emplea varios minutos para traducir un programa corto. Una vez completada la traducción, sin embargo, es necesario hacerlo otra vez. Una vez compilado el programa, puede cargarlo y ejecutarlo cuantas veces quiera, sin demoras. Es evidente que existen opciones. Se pueden llevar programas con intérpretes, con un seudocompilador o con un verdadero compilador. Si se usa un intérprete, la solución será rápida pero el programa se desarrollará más lentamente. Si se usa un compilador, el programa se desarrollará a mayor velocidad, pero cada vez que sea necesario efectuar una modificación habrá que darle casi tres minutos de tiempo al compilador. Si se usa un seudocompilador, se logra un compromiso entre la velocidad y la conveniencia.




¿Qué es mejor? ¿Un intérprete o un compilador? Todo depende del uso que quiera dársele.

Para decidir el tipo de traductor que debe elegirse, hay que pensar en el trabajo a realizar. Si es un problema rápido, puede ser preferible un intérprete. Seguramente resultará más conveniente escribir un programa con intérprete que con compilador: pueden corregirse errores y volver a pasar el programa de inmediato. Si se va a pasar una sola vez el programa, o a lo sumo dos veces, ¿a quién le importa demorar cinco minutos más en pasarlo?

De todos modos, si se está trabajando en un problema que requiere cada una de las fracciones de velocidad que puede rendir el ordenador, habrá que usar un compilador, que es también la elección correcta si se va a volver a pasar el programa una y otra vez, sin hacerle modificaciones. Cuando un programa está en su forma definitiva, tiene mucho sentido almacenarlo permanentemente.

Hay muchas situaciones en las que un buen compromiso entre compilador e intérprete consiste en desarrollar los programas con el segundo, para luego, una vez completados, pasados a limpio y funcionando satisfactoriamente, compilarlos para su rápida ejecución.

En la práctica, no habrá necesidad de elegir el tipo de traductor a utilizar. El propio sistema tendrá un intérprete o un compilador para el lenguaje que trabaje, aunque rara vez tendrá ambas cosas.

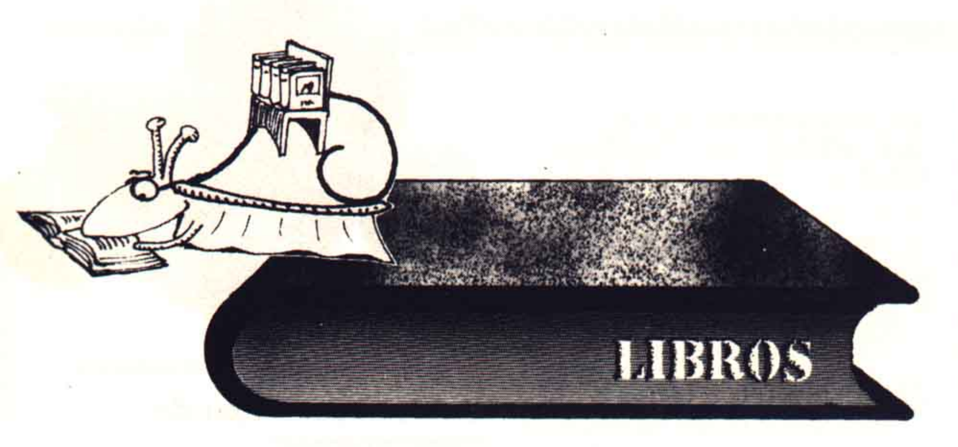


***Existen cuatro tipo de transcriptores; intérpretes, seudocompiladores, seudocompiladores incrementales y compiladores.***

## Conclusión

La mayoría de la veces, los diseñadores de lenguaje han hecho del *BASIC* un intérprete, del *PASCAL* un seudocompilador y de *FORTRAM* un auténtico compilador. Así, los programas en *BASIC* se ejecutan con ritmo bastante más lento que los escritos en otros lenguajes de alto nivel. No hay razón alguna para que estos lenguajes tengan que usar el tipo de traductores que usan. Podríamos tener un seudocompilador para *BASIC*, un compilador para *PASCAL* y un intérprete para *FORTRAN*. Es tonto comparar la velocidad de un lenguaje con la de otro, puesto que en gran parte la velocidad está determinada por el tipo de traductor que se use.

Dado que el *BASIC* se interpreta normalmente para propósitos prácticos, un programa *BASIC* irá más lento que uno en *FORTRAN* o en *PASCAL*. Pero recordemos, una vez más, que no se trata de una falta del *BASIC*. Cuando se elige un lenguaje interpretado, se está cambiando la velocidad por conveniencia.




largo, por lo que se disminuyen las posibilidades de cometer errores al introducirlos, además de facilitar al máximo su comprensión.

Resulta un libro adecuado para todos los estudiantes, a quienes no les gusta demasiado pasarse las horas delante de un libro y quieren darle una utilidad práctica a su ordenador.

**Título: MSX. El Manual Escolar**
**Autor: Voss**
**Editorial: Ferre Moret, S. A.**
**Páginas: 379**

De todos es conocida la utilidac de los ordenadores para eliminar extraterrestres o recorrer complicados laberintos. Pero existen otrc tipo de aplicaciones para las que el ordenador es igualmente válido. Una de estas aplicaciones es la educación.

*El Manual Escolar,* es un libro escrito para los alumnos de los últimos cursos de EGB y BUP, y contiene gran cantidad de programas que hacen que el estudio de determinadas materias, resulte una tarea más fácil, constructiva y amena.

Las materias tratadas en el libro son bastante variadas, dedicando un capítulo a cada tema específico, que son los siguientes: Matemáticas, Química, Física, Idiomas, Biología/Ecología, Contaminación ambiental, Geografía/Historia, Economía y Matemáticas II. Como se puede apreciar, los temas son muy variados y entretenidos, y ayudan a enfocar cualquier tipo de problema hacia el ordenador.

El libro comienza con un rápido repaso a las características del *BASIC* y a las instrucciones más empleadas. En capítulos posteriores se incluye el repaso a otras instrucciones menos utilizadas pero necesarias en algunas ocasiones. Los capítulos empiezan dando una descripción previa del problema y finalizan con el programa en cuestión. todos los programas están descritos de una for-




ma muy completa, siguiendo una estructura similar en todos los casos:

1. Presentación del problema,
2. análisis del problema.
3. diagrama de flujo,
4. codificación del programa,
5. descripción del programa y variables utilizadas, y
6. obtención de resultados.

Por otra parte, ninguno de los programas resulta excesivamente largo, por lo que se disminuyen las posibilidades de cometer errores al introducirlos, además de facilitar al máximo su comprensión.

**Título: LOGO de la tortuga a la inteligencia artificial**
**Autor: Luis Rodríguez-Roselló**
**Editorial: Vector Ediciones**
**Páginas: 581**

Es indudable que existe un interés creciente en todo el mundo por el lenguaje *LOGO.* Podría decirse que se trata de un fenómeno sociológico más que de un lenguaje de ordenador. Existen publicaciones periódicas dedicadas en exclusiva a este lenguaje, congresos dedicados íntegramente a *LOGO,* asociaciones de usuarios en muchos países, etc. A pesar de que todo el mundo hable de él, *LOGO* sigue siendo un gran desconocido, y la idea más generalizada sobre el mismo es que es un lenguaje para niños.

La incorporación de la informática a la enseñanza es un hecho imparable, y se diría que en la actualidad, es un fenómeno crítico en nuestro país. Es, por tanto, el momento idóneo para iniciar una reflexión que permita conocer a fondo *LOGO,* tanto desde un punto de vista informático, como de su filosofía educativa y sus posibilidades reales en la enseñanza. Este es el objetivo primordial que se propone el presente libro.

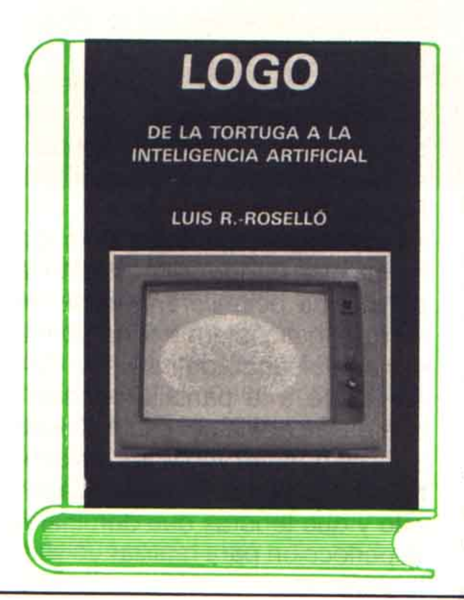
El enfoque dado a esta obra proporciona una visión general del *LOGO* sin centrarse en ningún dialecto concreto, procurando uti-

lizar aquellas primitivas que son comunes a la mayoría de las versiones del lenguaje, y haciendo hincapié sobre todo en las estructuras de los programas, alejándose de un enfoque de tipo «manual» para un ordenador concreto. El autor ha planteado acertadamente el libro, suponiendo que el lector ignora todo sobre este lenguaje, por lo cual comienza tratando los aspectos generales de ésta, su pedagogía asociada y su relación con el mundo de la informática en la educación.

A continuación comienza el aprendizaje concreto del lenguaje; se presentan los procedimientos incorporados, las entradas que necesitan estos procedimientos, la posibilidad de definir procedimientos el usuario, las variables, las listas, el control del editor y la sintaxis.

El siguiente capítulo nos introduce en la parte del *LOGO* más conocida, y quizá la más espectacular: los gráficos de tortuga. Aprendemos las instrucciones que permiten subir y bajar la pluma, mostrar y esconder la tortuga, situarla en coordenadas cartesianas, orientarla. También, abundan los ejemplos de cómo utilizarla, cómo por ejemplo, una serie de procedimientos para dibujar varios tipos de espirales utilizando un proceso recursivo.

Los gráficos de tortuga nos servirán también para comprender la mejor cualidad del lenguaje *LOGO:* la programación modular. Una serie de ejemplos nos muestran cómo diseñar módulos que realizan cada uno un dibujo, y luego combinarlos de forma que se interrelacionen.



# SOFTWARE

***Programa: North Sea Helicopter***
***Tipo: Juego***
***Distribuidor: Datamarket***
***Formato: Cassette***

Los simuladores de vuelo son, a veces, algo más que un simple juego. Este es un claro exponente de que la simulación, tiene como fin ayudar al aprendizaje, ya sea de vuelo en Helicóptero (como en este caso), o en avión, etc.

Sin embargo, no todo iba a ser volar, ya que, en *North Sea Helicopter,* se cuenta con el aliciente adicional de que a la vez que se intenta volar hay que rescatar a unos hombres que han sido arrojados al temible Mar del Norte por la tempestad.

Tu copiloto dará las coordenadas en donde se encuentra el único superviviente del desastre ocurrido en una plataforma petrolífera. Pero para cerciorarte de ello, pulsando la «M» aparecerá el mapa de la zona, donde se mostrarán la plataforma petrolífera, la base de tierra, el helicóptero y el hombre a salvar. Pulsando de nuevo la «M» se vuelve a la pantalla principal, que es la cabina del piloto. Desde aquí se pueden ver todos los controles y relojes que posee el aparato y que tendrás que dominar a la perfección para realizar un vuelo perfecto.

Desplázate hacia el lugar lo más rápidamente posible, pero con cuidado, ya que el tiempo se irá complicando bastante y llegará a ser de auténtica tempestad.

Tendrás que maniobrar de tal forma para acabar justo encima de él, que creerás hacer un auténtico vuelo en helicóptero. Una vez situado encima, pulsando la «R» se bajará una cuerda e izaremos el hombre a bordo. Esto sólo ocurrirá cuando nos encontremos en el lugar apropiado.

Desde el despegue, hasta el vuelo nocturno (con posibilidad de volar por infrarrojos), todo está pensado para lograr el máximo realismo, desde el sonido hasta la voz del copiloto, algo que podemos testimoniar es el mejor conseguido de todos los simuladores de vuelo que hemos visto.

El atractivo del juego es la necesidad de utilizar al menos un joystick, puesto que, como buen helicóptero necesita de dos mandos para ser maniobrado. El que tenga uno, tendrá que conectarlo en el port 1 de joystick y usar las teclas del cursor como joystick adicional.

Cada uno tiene una función específica, por lo que, si no tiene joystick será difícil jugar con él.

Las instrucciones son completas e indican todos los pasos a seguir para conseguir volar. La perfección sólo se consigue con la práctica.




***Puntuación:***
***Presentación: 8***
***Rapidez: 8***
***Claridad: 9***
***Adicción: 9***






**_Programa: Flight Deck_**
**_Tipo: Juego_**
**_Distribuidor: Datamarket_**
**_Formato: Cassette_**

Los incidentes del Golfo de Sidra se pueden plasmar en este programa directamente, donde la flota americana atacó Libia.

Es esta ocasión, tú estás al mando de un portaaviones situado cerca de una isla donde tienen su cuartel general unos terroristas. Tu misión consiste en desmantelar y destruir sus bases antes de que acabe el tiempo y sean ellos quien ataquen.

Hay tres pantallas diferentes. Con «F1» nos encontramos en el portaaviones, donde podremos controlar los aviones que tenemos, la velocidad y dirección del barco y el radar de corto alcance. La tecla «F2» nos da el mapa, en el que veremos la posición relativa del barco y los aviones con respecto a la isla y el combustible de los aviones en vuelo; y «F3» nos da una imagen de la isla y del combustible de los aviones.

En el portaaviones, posees 10 tipos de aviones, entre cazas, bombarderos y aviones de reconocimiento. Con los cazas has de derribar cuantos aviones enemigos aparezcan en la isla. Con los aviones de reconocimiento tendrás que fotografiar la isla completa para descubrir las bases escondidas y con los bombarderos tienes que destruir sus bases y escondites. La puntuación se realiza en base al número de aviones que te quedan, la cantidad de isla fotografiada y en tu habilidad para destruir cazas enemigos y aterrizar en el portaaviones.

Todo esto suena sencillo y fácil, pero nada más lejos de la realidad. Para empezar has de controlar el barco y situarlo de forma que el viento no cruce la cubierta de despegue, pues esto resulta fatal para los aviones. Con las teclas del cursor arriba/abajo, controlaremos la velocidad del portaaviones, y con las teclas derecha/izquierda, el desplazamiento del barco. Pero ¡cuidado!, si te acercas demasiado a la isla, los aviones enemigos atacarán al barco hasta que lo hundan o hasta que salgas de la zona peligrosa que rodea la isla. De todas formas, con tus cazas podrás contrarrestar el ataque.




Una vez situado el barco en el lugar ideal, elige el tipo de avión que quieres utilizar. Esto se hace sacando uno a uno del hangar y trasladándolo con el tractor hacia uno de los tres lugares existentes en la cubierta o bajándolos mediante el ascensor para colocarlos a la cola, ya que los diez aviones están en una fila en hangares sucesivos. Cuando uno sale a cumplir una misión, el resto de los aviones se desplazan hacia el lugar libre. De esta forma, el avión que está en vuelo, si consigue aterrizar (maniobra ciertamente peligrosa y difícil) se sitúa al final de la fila y espera su turno.

El tractor y los aviones en vuelo se controlan con joystick. El despegue y el aterrizaje requieren habilidad y mucha práctica. De manera que no os preocupéis si al principio no sale bien.

**_Puntuación:_**
**_Presentación: 9_**
**_Rapidez: 8_**
**_Claridad: 8_**
**_Adicción: 8_**

# SOFTWARE

**Programa: Beamrider**
**Distribuidor: Proein, S.A.**
**Tipo: Juego**
**Formato: Cassette**

Un nuevo tipo de juegos espaciales está apareciendo. El espacio sideral se ha quedado atrás, las luchas más despiadadas ocurren en olvidadas Galaxias perdidas y entre los más extraños seres, pero vivamos en un lugar u otro, seamos guerreros o contrabandistas, siempre seremos atacados porque estamos en la Era de la supervivencia.

*Beamrider* hará que te encuentres en una estación espacial, de la que estás a punto de salir porque una peligrosa misión va a comenzar. Contarás con toda clase de armas sofisticadas, como un rayo láser que fulminará a tus adversarios y torpedos que utilizarás en caso de encontrarte una fotaleza o nave enemiga, objetivo de tu misión.

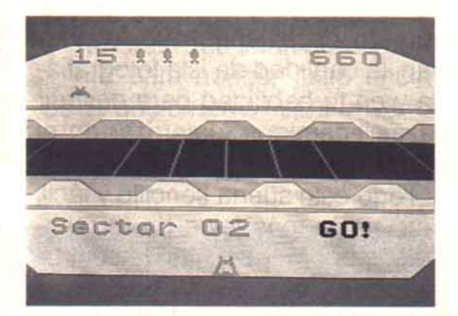
Pero a parte de esto —que te proporcionará una gran puntuación— debes exterminar todas las naves enemigas que encuentres a tu paso.

Velocidad de reacción será una de tus grandes armas a la hora de destruir a tu enemigo, pero que no te sirva de descuido, ya que existen objetos inmóviles que se estrellarán contigo si no controlas bien tu nave.

Irás avanzando en sectores planetarios cada vez que consigas batir a quince platillos voladores blancos, así irás consiguiendo puntos y podrás acceder a destruir la nave nodriza, pero sólo podrás atacarla con un torpedo, así que no los desperdicies.

A primera vista el juego se corresponde en todas sus facetas a los característicos juegos de marcianitos, pero también debemos darnos cuenta que el espacio es siempre del mismo color. Las diferencias de este juego no se encuentran en el tema, ni en su ejecución, sino en la realización, tanto gráfica como en sonido.

*Beamrider* parte de una nave nodriza en la que podemos decretar nuestra hora de despegue, las compuertas se abrirán, acompañadas de un potente sonido, que nos anuncia la hora de la batalla, sirve tanto para alertarnos como para concentrarnos. La supervivencia a comenzado, nuestro radar empieza a detectar intrusos, las coordenadas de posición nos serán dadas gracias a la nueva concepción del espacio. Una perspectiva creada hacia el infinito




por unas rayas fluorescentes y la continua sucesión de horizontales hacia nosotros, hace que la velocidad y el acercamiento a las naves enemigas se proyecten sin darnos cuenta, en una distracción más que agilizará en gran medida nuestros reflejos.

Platillos, naves, rastreadores, minas y escombros serán los objetos de destrucción que constantemente nos harán perder la paciencia, pero recuerda la ley de la supervivencia «matar para que no te maten», aunque esto no será necesario porque, a diferencia de otros juegos, si ves que pierdes tu concentracióm y no puedes seguir, puedes pulsar el cero y obtendrás una tregua.

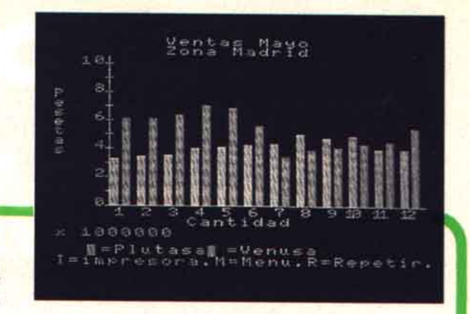


Otra característica a destacar del juego es la posibilidad de dejar de jugar en una partida de tres o más jugadores, sin tener que dejar de jugar estos, cuando sea tu turno y decidas retirarte pulsa el número cinco y tu aventura habrá finalizado.

Una advertencia: en los sectores más altos, como puede ser el 10, no te dará tiempo a fijarte en si has conseguido destruir las naves que aparecen a tu encuentro, por eso debes aprender a guiarte del sonido y los cambios de color de tu pantalla, sin duda pasarás momentos de tensión, pero tienes una misión que cumplir, ¡adelante!

***Puntuación:***
***Presentación: 8***
***Claridad: 6***
***Rapidez: 7***
***Adicción: 7***






**Programa: Gráficas de Gestión**
**Distribuidor: Idealogic**
**Tipo: Aplicación**
**Formato: Cassette**

Este programa ha sido realizado para que Ud. pueda «ver» en forma gráfica los resultados de su gestión, sus cuentas o sus cálculos.

Es necesario para realizar una buena gestión, conocer desde el principio todos los datos con los que cuenta a la hora de preparar un informe o tomar una decisión. Seguir un detallado estudio sobre las posibilidades, aumentos o disminuciones en un determinado período, es la base de un acertado comportamiento ante los cambios eventuales que puede sufrir una economía, un producto e incluso una persona (pirámides poblacionales).

La estadística siempre ha sido una ciencia que se ha visto un poco devaluada en sus posibilidades, porque se basa en probabilidades, pero de todos es sabido que a la hora de tomar una decisión o efectuar un cambio, la estadística marca la pauta a seguir en el comportamiento del fenómeno.

*Gráficas de gestión* es un programa para nuestro gusto muy recortado, porque el primer requisito que se espera al contemplar un gráfico de gestión es la singularización en un golpe de vista la situación financiera de una empresa o el desarrollo obtenido a lo largo de un período.

Para estos requisitos nuestro programa está suficientemente capacitado, pero no cuenta con lo que podríamos llamar memoria, algo totalmente necesario para poder comparar gráficos, es decir, si quisiéramos observar la trayectoria económica de dos bienes complementarios al mismo tiempo, no podríamos, pues tendríamos que verlos individualmente.

Vamos a hacer referencia a los apartados con los que cuenta y comentarlos uno por uno: Entrada de datos. Aquí se le permite definir 3 columnas con 12 datos por cada una de ellas, bien por meses o numerados del 1 al 12.

Modificaciones y Listado. Le permite modificar (datos o título), además de introducir nuevos datos e imprimir.

Rotulación. Permite nominar a sus gráficas, ejes (x e y) y el período de trabajo.

Gráficas de barras, lineal y circular. Estos son tres apartados diferentes para elegir el método más cómodo de representación. En estos apartados podemos hacer notar la falta de color en la gráfica de sectores, ya que los tanto por ciento quedan bien delimitados, pero sería mejor si a cada dato se le hubiera designado un color. Sería más inmediata la observación.

Grabación y Lectura. Esta opción se subdivide a su vez en varias posibilidades de obtención de listados o almacenamiento en cassette.

En el segundo apartado de Lectura, debe saber que una vez leido el fichero, si opta por introducir nuevos datos, los anteriores desaparecen.

Códigos de impresora. Aquí se le permite definir la densidad gráfica «tamaño», de la gráfica a imprimir. Si Ud. posee una impresora para tratamiento de pantallas gráficas, cambiando el código podrá obtener, *Bit Image Graphis,* pasando de la simple resolución a la pantalla gráfica de mayor resolución.

Notas complementarias, el editor le facilitará en todo momento lo que debe realizar, siga sus instrucciones. Una advertencia: nunca pulse la opción Impresora ya que perdería el control sobre el programa si no está conectada ésta.

Es un programa para realizar gráficas a nivel muy personal y sobre temas muy individuales, puesto que si las gráficas se quieren elaborar como datos finales, hay que hacer un estudio previo en el que se reflejen claramente los datos de estudio muy sintetizadamente.

Con esto no queremos desvirtualizar la propiedades del programa, sino simplemente hacerle saber sus posibilidades.

# Programas Sony MSX, para lo que guste ordenar.







JUEGOS

GESTION







EDUCATIVOS

APLICACIONES






# PROGRAMAS SONY MSX

**Educativos**

- Monkey Academy
- Alfamat
- Viaje Espacial
- Multipuzzle
- Noria de Números
- Corro de Formas
- Coconuts
- Yo Calculo
- Selva de Letras
- El Cubo Informático
- Electro-graf
- El Rancho
- Teclas Divertidas
- Boing Boning
- Compulandia
- Mil Caras
- Logo
- Países Mundo-1
- Países Mundo-2
- Tutor
- Computador Adivino
- Aprend. Inglés-1
- Aprend. Inglés-2
- Cosmos
- Curso de Básic
- Juego de Números

**Juegos**

- Antártic Adventure
- Athletic Land
- Sparkie
- Juno First
- Car Jamboree
- Battle Cross
- Crazy Train
- Mouser
- Computer Billiards
- Alí Babá
- Track & Field-I
- Track & Field-II
- Dorodon
- Chess (Ajedrez)
- Senjo
- E.I.
- Lode Runner
- Super Tennis
- Backgammon
- Super Golf
- Hustler
- Binary Land
- Driller Tanks
- Stop the Express
- Ninja
- Les Flics
- La Pulga
- The Snowman
- Cubit
- Pack 16K
- Fútbol
- Kung Fu
- Batalla Tanques
- Mr. Wong
- Xixolog
- Buggy
- Sweet Acorn
- Peetan
- Jump Coaster
- Buggy 84
- 3D Water Driver
- Pinky Chase
- Wedding Bells
- Fightting Rider

**Aplicación**

- Memoria Ram 4 K
- Creative Greetings
- Character Collect
- Quinielas y Reducciones
- Pascal
- Ensamblador
- Generador Juegos

**Gestión**

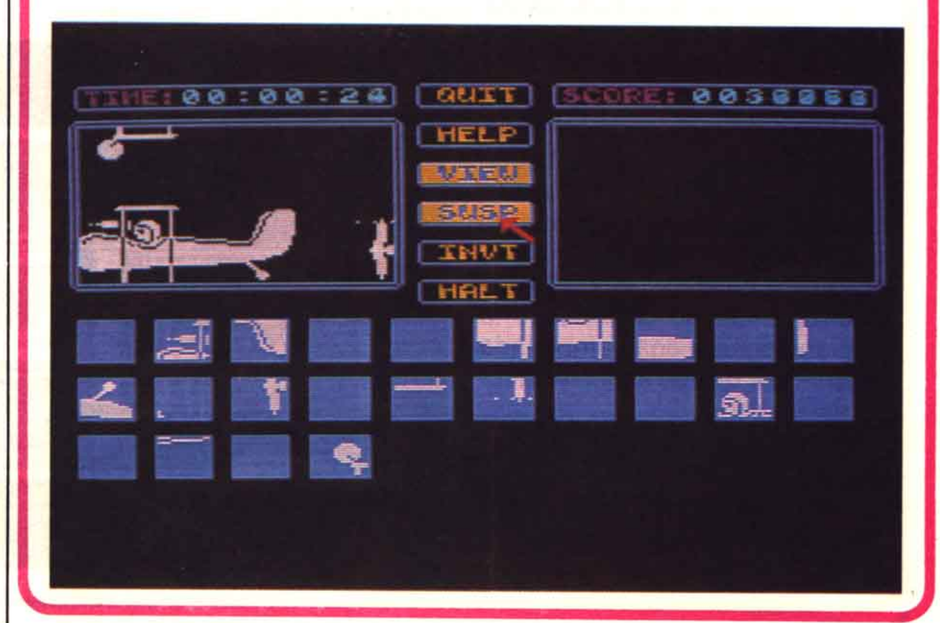
- Hoja de Cálculo
- Homewriter
- Control Stocks
- Contabilidad Personal
- Ficheros
- Procesador de Textos
- Control Stocks
- Vencimientos
- Contabilidad 1.500

# SOFTWARE



***Programa: Confused?***
***Tipo: Juego***
***Distribuidor: Datamarket***
***Formato: Cassette***

Los rompecabezas siempre ha sido la forma de entretenimiento más popular. ¿Quién no ha resuelto, al menos una vez, un puzzle? Los hay de mayor o menor dificul-





tad, así como de mayor o menor número de piezas. Estos puzzles presentaban un problema de espacio bastante importante, si eran muy grandes había que ponerlos en una mesa o en una gran tabla para acabarlos.

Esto ya no es problema cuando se trata de utilizar el ordenador. Además, este programa es el primero en su especie para resolver rompecabezas.

En la pantalla aparecen dos ventanas y un menú con las distintas opciones. En la ventana derecha, resolveremos el rompecabezas y en la izquierda, bajo el logo del juego se esconde el dibujo original que hay que realizar.

Contamos con la posibilidad de escoger entre diez dibujos distintos, así como la dificultad del dibujo en cuestión. Si se escoge el más pequeño (muy sencillo de resolver) que es 2 x 2, el dibujo se dividirá en cuatro partes que tendrá que colorar en el orden adecuado en el menor tiempo posible y siempre antes de que se te acabe, puesto que a mayor dificultad, mayor es el tiempo que tienes para resolverlo.

Las piezas del puzzle se mueven con la ayuda de las teclas del cursor y la barra espaciadora o un joystick. Con ellos se cogen las piezas y se van situando en los lugares adecuados. Si lo ha hecho bien, es decir, si ha acertado la situación y la pieza al completar el puzzle sonará una musiquilla y aparecerá en la ventana de la izquierda el dibujo original, de lo contrario sonará una campanada indicándole que se acabó el tiempo asignado.

En principio parece sencillo, pero la característica principal de estos rompecabezas, el movimiento, complican mucho la resolución de éstos.

Tienes ayudas para resolver los problemas, pero éstas restarán puntos. El límite de tiempo está prefijado en función del número de piezas del rompecabezas, siendo el tiempo inicial cuando el puzzle es de 2 x 2 menor que cuando es de 12 x 8.

Prueba tu paciencia y habilidad para resolver todos y cada uno de estos rompecabezas, en el menor tiempo posible.

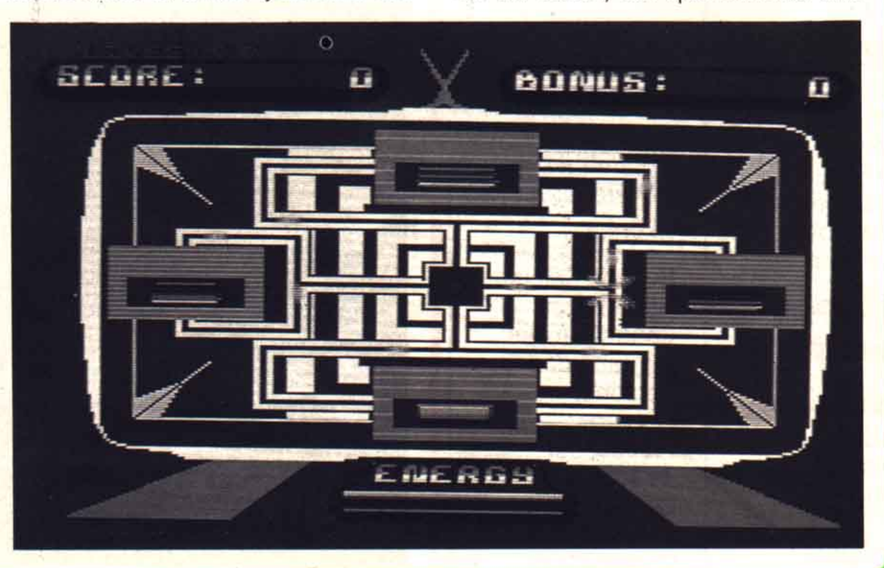
***Puntuación:***
***Presentación: 8***
***Rapidez: 10***
***Claridad: 9***
***Adicción: 8***

***Programa: Drome***
***Tipo: Juego***
***Distribuidor: Dinamarket***
***Formato: Cassette***

*Percy* se ha introducido en el corazón de *Drome*. Ayúdale a realizar su peligrosa misión, que es la de destruir el sistema tan completo y potente de *Drome*.

Se trata de un super ordenador descontrolado, el cual intentará eliminarte. Por suerte, tu odisea se inicia en la sala de control central de *Drome*, de donde tendrás que partir hacia uno de los cuatro sectores de que se compone y destruir todos y cada una de las defensas que tiene. La misión es complicada, pero estás tú para echarle una mano, que falta le hace.

Uno de los sectores es el interior de un cable, del que emana una





fuente de electrones y protones que intentarán atacarle. Obviamente tendrás que destruirlos con tu cañón de neutrones.

Elimina el máximo número de ellos para desactivar esta defensa.

Otro sector es el de las motos de luz. Una de estas motos es la del ordenador, que intentará hacerte estrellar con la estela que deja la tuya. Procura esquivarla a la vez que intentas destruirla de la misma manera, acelera, despégate de ella y enciérrala para que se estrelle.

El sector de los tanques es el más peligroso, puesto que éstos van detrás de tí a la vez que disparan sus cohetes. Aguarda tu oportunidad y escóndete. Llegará tu momento y cuando lo haga, dispara a matar.

Por último, está la habitación de la válvula principal. Aquí se trata de destruir una sección de la válvula y escapar a través de ella. Este lugar se halla protegido por múltiples cañones de rayos láser, que barrerán el suelo para localizarte y destruirte. Sólo con paciencia y perseverancia conseguirás tu propósito. Dispara continuamente, a la vez que esquivas sus rayos, y frena a la super máquina.

***Puntuación:***
***Presentación: 7***
***Rapidez: 8***
***Claridad: 7***
***Adicción: 7***

# Los modos de pantalla



***Tras haber concluido el estudio de los 4 sistemas de almacenamiento de la memoria de vídeo y de las figuras móviles, pasamos a estudiar en este número las transiciones entre las tablas de los distintos modos de pantalla.***

Suponemos que habréis compendido bien el funcionamiento de las tablas que componen los cuatro sistemas de almacenamiento. Pero esta serie pretende profundizar aún más en esas tablas, comentando los valores iniciales que toman y observando los cambios producidos en ellas al alterar el sistema de almacenamiento en el que estamos trabajando.

## Enchufemos nuestro ordenador MSX

El primer modo del que podéis disponer al enchufar vuestro ordenador es el modo *SCREEN 0,* cuyo sistema de almacenamiento sólo posee las tablas 0 y 2. La tabla 0, encargada de la distribución en pantalla de los carácteres, puede tomar dos valores iniciales. Al conectar el ordenador, la pantalla muestra el mensaje "*MSX BASIC* Version 1.0. Copyright 1983 by Microsoft. 28815 *bytes free.*" (en el caso de que tu ordenador sea de 16 K de memoria, pondrá "12431 *bytes free.*") y también muestra en la parte baja de la pantalla el contenido de las teclas de función, pues el ordendaor tiene en su interior una zona de la memoria normal (no de la de vídeo) que ordena cómo debe disponerse la tabla 0 al enchufar. Pero si desde esa situación introduces un comando *SCREEN* ordenando situar otro modo de pantalla, y luego vuelves al modo *SCREEN 0* de la misma manera, la tabla 0 borrará la pantalla, manteniendo el número de columnas que hubiera la última vez que usasteis el modo *SCREEN 0* (si no habéis utilizado la orden *WIDTH* desde que lo conectásteis, mantendrá un *WIDTH 37*) y situará el contenido de las teclas de función si no habéis ejecutado un *KEY OFF.* Pero lo importante para el programador es que la tabla 0 borra la pantalla SIEMPRE que hay un cambio de modo de pantalla, luego si tenemos un texto en la pantalla en el modo *SCREEN 1,* y queremos mantenerlo en el modo *SCREEN 0,* tendremos que volverlo a pintar. Por lo tanto el comando *SCREEN* también puede sustituir a la orden *CLS,* si es que nos hallamos en el modo *SCREEN 0.*

La tabla 2, encargada de almacenar los patrones de los caracteres, al ejecutarse la orden *SCREEN 0* toma los patrones de la lista de los 256 caracteres MSX. Esto es muy peligroso si habéis introducido nuevos caracteres, pues estos se borran y son sustituidos por los caracteres originarios. Por lo tanto, salvaguardad los datos de vues-



el modo *SCREEN 0,* nada más enchufar el ordenador, la zona libre de memoria está llena de datos que poseen los ordenadores SONY, y como para pasar a *BASIC* ha de aparecer en la pantalla un MENU, con el nombre del ordenador trazado encima, y todo ello en el modo *SCREEN 2,* los datos de ese sistema de almacenamiento se mantienen al pasar al modo *SCREEN 0,* exceptuando las zonas ocupadas por las tablas 0 y 2. Así os podréis encontrar las 3 figuras móviles que el banco de datos utiliza, diseñadas entre los octetos 14336 y 14359, en lo que en el modo de pantalla era la tabla 14.

En esa zona libre de la memoria podéis almacenar los datos que queráis, e incluso podéis utilizarla para almacenar números que tengáis que usar en vuestros programas, o grandes bloques de datos.

tros caracteres en otra parte del ordenador, si es que queréis volverlos a utilizar.

Pero la tabla 0 tiene 960 octetos, y la tabla 2 tiene 2048, y sin embargo la memoria de vídeo tiene un total de 16384. ¿Qué sucede en el sistema de almacenamiento del modo *SCREEN 0* con los 13376 octetos restantes? Pues sucede que simplemente se quedan vacíos, o mejor dicho, que no se alteran. Si hemos diseñado en el modo *SCREEN 1,* por ejemplo, unas figuras móviles y las hemos introducido en la tabla 9 (comprendida entre los octetos 14336 y 16383), al pasar al modo *SCREEN 0* los patrones no se borran, y se mantienen en la memoria de vídeo inalterados, pues esa zona de la memoria de vídeo es zona libre en *SCREEN 0* (los 13376 octetos libres están distribuidos en nuestros ordenadores MSX de la manera siguiente: son los comprendidos entre el 960 y el 2047, y los comprendidos entre el 4096 y el 16383. En la figura 1 podéis ver más detenidamente las zonas libres de la memoria de vídeo en los cuatro sistemas de almacenamiento).

Si alteráis un valor en los octetos de la zona libre a partir del 14336, ese valor alterado se mantiene luego al pasar a otros modos de pantalla, luego SE PUEDEN DISEÑAR SPRITES EN EL MODO *SCREEN 0,* aunque no se puedan utilizar más que en los otros modos de pantalla. A esto nos referíamos en el número anterior cuando hablábamos de que también se puede establecer una identificación entre los patrones de los caracteres y los patrones de figura móvil en el modo *SCREEN 0.*

Si poseéis un ordenador SONY MSX, os daréis cuenta de que en

En el sistema de almacenamiento del modo *SCREEN 1* existen las tablas 5, 6, 7, 8 y 9. La tabla 5 funciona exactamente igual que la tabla 0 en el sistema de almacenamiento del modo *SCREEN 0,* y la tabla 7 funciona igual que la tabla 2 con los patrones de los caracteres. La tabla 6, encargada de los colores en el modo *SCREEN 1,* al ejecutarse la orden *SCREEN 1* inicializa todos los caracteres con los colores establecidos de fondo y tinta, poniendo a todos los caracteres con la misma combinación de colores. Aunque hubiéramos alterado alguna combinación, la inicialización de la tabla 6 es drástica.

La tabla 8, al igual que las tablas 13 y 18, no cambia nada al ejecutarse la orden *SCREEN 1,* a excepción de la coordenada «y» de todas las figuras móviles, que toma el valor 209, situando a todos los

«*sprites*» fuera de la zona visible de la pantalla. Al enchufar el ordenador, cada plano de proyección contiene a la figura móvil de su mismo número (el plano 0 la figura 0, el 1 la 1, etc...) y si son figuras móviles «grandes», el plano 0 contiene a la 0, el 1 a la 4, el 2 a la 8, etc... (como ya vimos en el capítulo quinto, dedicado a las figuras móviles).

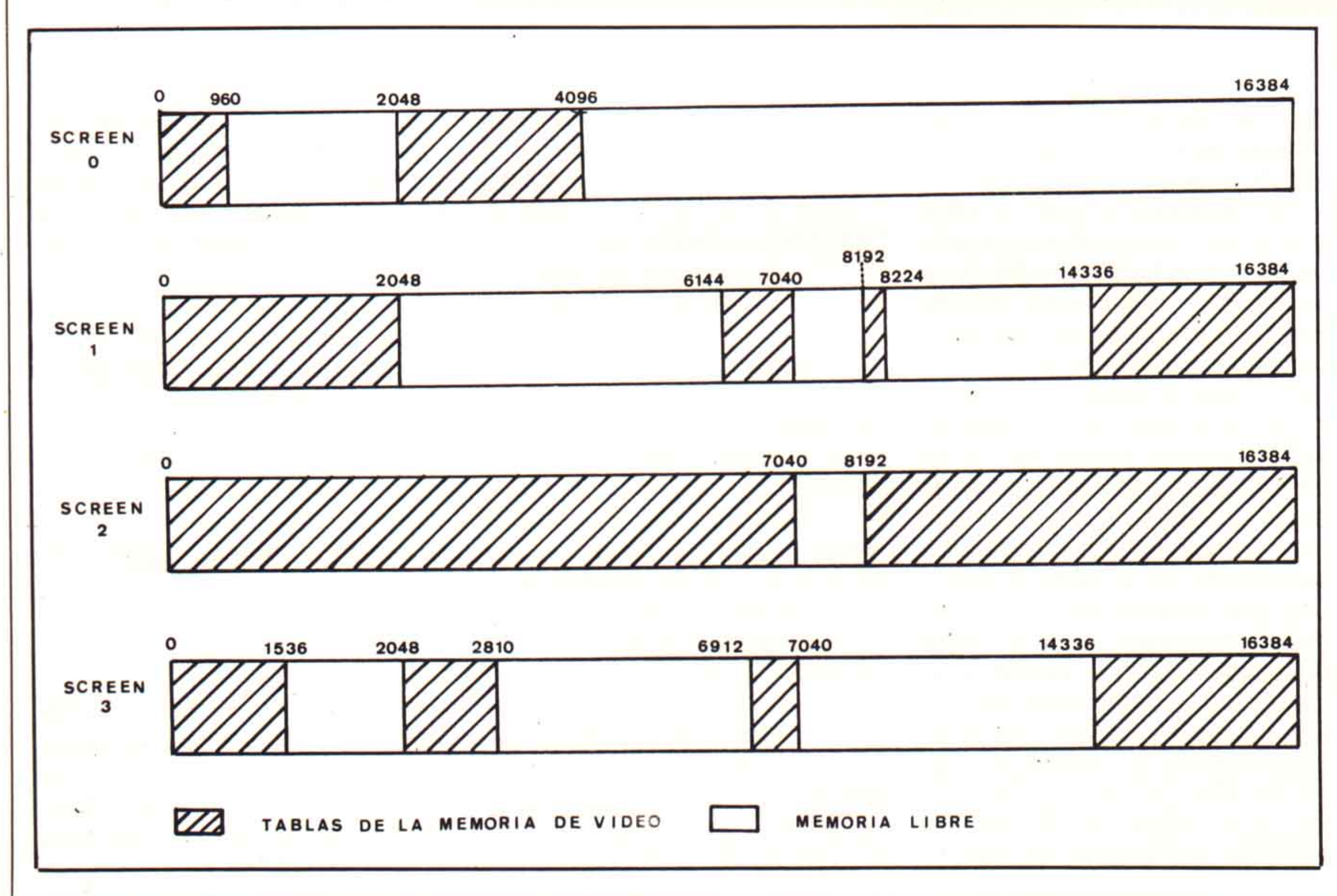
La tabla 9, como las tablas 14 y 19, no sufre ningún tipo de cambio al ejecutarse el comando *SCREEN,* sea cual sea el modo en el que nos hallemos.

El sistema de almacenamiento del modo *SCREEN 1* tiene un total de 11360 octetos de memoria libre, distribuidos de la siguiente manera: la zona de memoria comprendida entre el octeto 2048 y el 6143, la zona entre el 7040 y el 8191, y la zona entre el 8224 y el 14335.

En el sistema de almacenamiento del modo *SCREEN 2,* la tabla 112, responsable de los *pixels* que han de salir de color de tinta o de color de fondo, aquella tabla a la que llamábamos «mapa de la pantalla», al ejecutarse la orden *SCREEN,* borra absolutamente todos los datos que tenía e introduce ceros en todos sus octetos. La tabla 10, que situaba unos cuadros en función de otros del mismo campo, al ejecutarse la orden *SCREEN 2* sitúa cada cuadro en función de sí mismo. El manejo de estas dos tablas, en consecuencia, ha de hacerse con cuidado, pues si tenéis algo dibujado en el modo *SCREEN 2* se os borrará al cambiar de modo de pantalla, y lo mismo sucederá con los trucos gráficos que hayáis logrado pintar con la tabla 10.

La tabla 11 inicializa a TODOS sus octetos con el color de fondo establecido por la última orden *COLOR,* y a todos les sitúa un color de tinta transparente (color número 0). En el momento en que desde *BASIC* queramos impregnar de un color a un *píxel* con la orden *PSET,* y no especificamos el color, el octeto correspondiente de la tabla 11 adquiere el último color de tinta especificado en una sentencia *COLOR.*

Estas drásticas maneras de inicializarse que tienen estas tablas al ejecutarse la orden «*SCREEN 2*», es decir, al producirse la transición de un modo de pantalla cualquiera al modo *SCREEN 2,* en el caso de que usar este modo sea imprescindible (en muchos casos



lo es). El sistema de almacenamiento del modo *SCREEN 2* sólo tiene 1152 octetos de memoria libre entre los octetos 7040 y 8191, ambos inclusive. (Aún así, luego veremos un truco gracias al cual sí se pueden conservar los datos de las tablas del modo *SCREEN 2*).

La tabla 17, al ejecutarse la orden *SCREEN 3,* introduce en todos los bloques de 4 x 4 *pixels* el color de fondo que esté establecido, luego destruye los dibujos que estén en la pantalla. La tabla 15, encargada de situar unos cuadros en función de otros, también se inicializa drásticamente, poniendo cada cuadro en función de sí mismo, y haciendo desaparecer, como sucedía en el sistema de almacenamiento del modo *SCREEN 2,* todos los trucos gráficos que hubiera pintados.

El modo *SCREEN 3* tiene 11904 octetos de memoria libre, distribuidos de la siguiente manera: los octetos comprendidos entre los octetos 1536 y 2047, entre el 2916 y el 6911, y entre el 7040 y el 14335.

## ¿Qué hacer con la memoria libre? Aplicaciones

La principal aplicación de la memoria de vídeo libre es, por supuesto, llenarla de datos que sean necesarios para vuestros programas, pero teniendo siempre cuidado con los procesos de transición de un modo de pantalla a otro, pues es posible que el otro modo de pantalla al que queréis pasar ocupe esa memoria libre con tablas que se inicializen de forma drástica, con lo cual esos datos que teníais almacenados se han borrado, y tenéis que volverlos a introducir. Fijaos bien en la figura 1, que representa las zonas libres de la memoria de vídeo en los cuatro sistemas de almacenamiento, antes de elaborar un programa que conlleve un proceso de transición de un modo a otro. Para manejar la figura 1 debéis tener mucho cuidado, pues en este mapa de la memoria hemos incluido los números de los octetos que comienzan una zona, pero no los que la terminan, así que, por ejemplo, si tomáis la zona de memoria que va desde el octeto 2048 all 6144 en el sistema de almacenamiento del modo *SCREEN 1,* esa zona es en realidad la que va del 2048 al 6143, ambos inclusive, pues el octeto 6144 ya pertenece a la zona siguiente. Tened, por lo tanto, en cuenta, que una zona siempre comienza con un número par y termina con uno impar.

Pero la memoria libre no sólo sirve para almacenar datos arbitrarios, sino también para trasladar datos dentro de la misma memoria de vídeo.

Un ejemplo bastante sencillo de esto último es la traslación de caracteres programables. Como ya dijimos antes, los caracteres establecidos en el sistema de almacenamiento del modo *SCREEN 0* en la tabla 2, se pierden al pasar al modo *SCREEN 1,* pues la tabla 7 inicializa con los 256 caracteres MSX, y no considera si habéis cambiado algún carácter o no. Sin embargo, los datos de vuestros caracteres cambiados se conservan donde estaban, es decir, entre los octetos 2048 y 4095, que en *SCREEN 1* son de memoria libre. Tenéis la suerte, por lo tanto, de que vuestros datos no hayan sido borrados, a pesar de que los caracteres que van a escribirse en la pantalla son los que se encuentran en la tabla 7, entre los octetos 0 y 2047. Si ahora introducía la orden:

```
FOR T=0 TO 2047:VPOKE
T,VPEEK(T + 2048:NEXT
```

en realidad estáis situando en la tabla 7 todos los datos que antes se hallaban en la tabla 2, y por tanto se conservan vuestros caracteres programados, y salen a la pantalla en el modo *SCREEN 1.* Para realizar el proceso inverso, no podéis hacer lo mismo, pues en el sistema de almacenamiento del



modo *SCREEN 0* no son memoria libre los octetos del 0 al 2047, porque la tabla 0 ocupa 960 octetos, del 0 al 959. De ese manera, al cambiar al modo *SCREEN 0,* buena parte de vuestros datos han desaparecido y no podéis introducirlos en la tabla 2. Para poder hacerlo tenéis que, todavía en el modo *SCREEN 1,* desplazar la tabla 7 completa hacia alguna zona que sea de memoria libre en ambos modos, como la zona comprendida entre los octetos 4096 y 6143, o como la inmensa zona que va desde el 8224 al 14335. Una vez desplazada, se ejecuta la orden *"SCREEN 0"* y entonces los datos de la tabla 7 han de ser «redesplazados» a la tabla 2, y allí podrán

# el mejor softwa







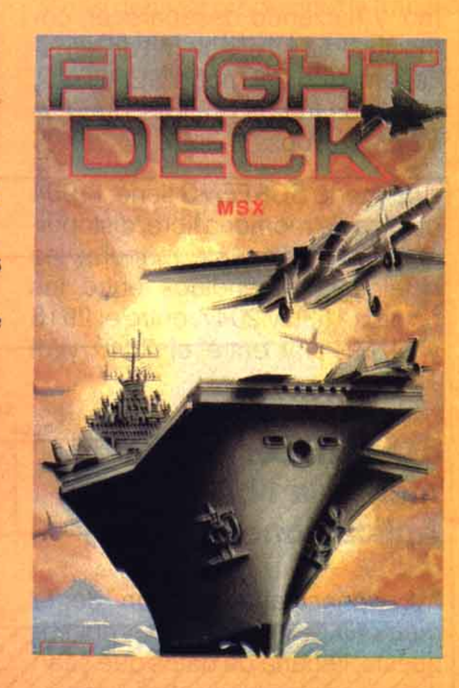
**DROME**

Entretanto en DROME, un Super-ordenador, debes encontrar y eliminar los sofisticados sistemas de defensa y supervivencia.

Has de elegir uno de los cuatro sectores que constituyen los mecanismos de defensa de esta terrorífica máquina.

Un atractivo juego de acción, donde se pone a prueba la capacidad de la máquina y del jugador.

**Precio de venta 2.600 ptas. (IVA incluido)**




**FLIGHT DECK**

Sienta la emoción del golfo de Sidra en casa.

FLIGHT DECK es un juego de estrategia y habilidad en el que tendrás que desmantelar las bases enemigas.

Al mando de un portaaviones donde dispones de 10 unidades de combate... y poco tiempo.

**Precio de venta 2.600 ptas. (IVA incluido)**







**MC-ATTACK**

Ayuda a Fredy, el Rey de la Hamburguesa a preparar el suculento manjar que hace las delicias de los comensales.

Ten cuidado con las salchichas grasientas y los huevos escurridizos que intentarán arruinar tu exquisito plato.

Defínete con la pimienta y procura hacer el mejor número de hamburguesas posible.

... Buen provecho.

**Precio de venta 1.900 ptas. (IVA incluido)**

# re para MSX




## CONFUSED?

Es el puzzle electrónico.

El objeto del juego es resolver 10 puzzles con distinto número de piezas, a elegir, pero todas... MOVIENDOSE.

Pon a prueba tu inteligencia y capacidad de deducción para solucionar algunos de estos entretenidos rompecabezas.

**Precio de venta 2.600 ptas. (IVA incluido)**

## NORTH SEA HELICOPTER

Una explosión en una plataforma en el mar del Norte arroja a los hombres a un destino incierto...

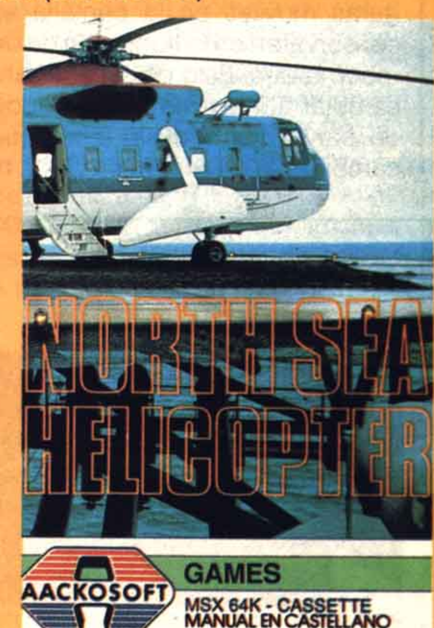
Empieza una carrera contrareloj para salvarles de su fatal situación.

Tienes que convertirte en un piloto experimentado para mantener el control del helicóptero... El tiempo empeora.

¿Crees que cumplirás la misión?

**Precio de venta 2.600 ptas. (IVA incluido)**







## SPACE RESCUE

Desciende de tu nave nodriza a la superficie lunar e intenta rescatar a los hombres extraviados.

Ojo con los meteoritos que te destruirán cuando intentes regresar.

Disponer de nuevas plataformas para culminar con éxito la misión.

Desgraciadamente tu nave nodriza está bloqueada por unidades de combate enemigas... Intenta destruirlas.

**Precio de venta 2.600 ptas. (IVA incluido)**

**infodis, s.a.**

Solicita tu programa favorito en tu tienda especializada habitual o recorta y envía este cupón a INFODIS, S. A., c/Bravo Murillo, 377. 5.º A - 28020 MADRID.

NOMBRE ______
DIRECCION ______
POBLACION ______ PROVINCIA ______ C. P. ______
Forma de pago: CHEQUE ☐ CONTRAREEMBOLSO ☐
TARJETA DE CREDITO: American Express ☐ VISA ☐ INTERBANK ☐
Número de mi tarjeta ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
CADUCIDAD ______
Firma ______

| TITULO | CANTIDAD | TITULO | CANTIDAD |
|---|---|---|---|
| DROME<br>FLIGH DECK<br>MC-ATTACK | | CONFUSED?<br>NORTH SEA<br>HELICOPTER<br>SPACE RESCUE | |

# La pantalla

salir vuestros carácteres a la pantalla.

Algo parecido sucede con las figuras móviles, cuyos patrones se mantienen en la memoria en cualquier modo *SCREEN,* pero que al cambiar de modo de pantalla son pintadas fuera de la zona visible. Si desplazáis la tabla 8 (o la 13 ó la 18) a la zona comprendida entre los octetos 7040 y 7167, zona que pertenece a la memoria libre en todos los sistemas de almacenamiento de los cuatro modos *SCREEN,* y a continuación «redesplazáis» esta tabla a su posición inicial en otro modo de pantalla, los datos de la posición de las figuras móviles en la pantalla se conservarán en la transición de un modo a otro. Esto os servirá para cambiar libremente de un modo de pantalla a otro sin tener que cambiar para nada (ni en color, ni en patrón, ni en posición) las figuras móviles que estéis usando.

Otra de las aplicaciones de la memoria de vídeo libre tiene que ver con la otra memoria, la memoria que vosotros utilizáis al programar, al apretar una tecla, al volver de una subrutina, etc... Esa memoria ha sido tratada ya en bastantes ocasiones por *MSX Magazine,* y ya sabréis mucho de ella. Sólo os recordaremos que contiene la *ROM* y alberga también la *RAM,* y que en esta última se almacenan vuestros programas en *BASIC.*

Si vuestro ordenador MSX es de los llamados «pequeños», y al enchufarlo sale en la pantalla el mensaje "12431 *bytes free*", que significa "12431 octetos disponibles" (refiriéndose, por supuesto, a la *RAM,* no a la memoria de vídeo, que tiene 16384 octetos en todos los modelos MSX) entonces la aplicación de la memoria libre que a continuación explicamos no puede realizarse en vuestro ordendaor. Si vuestro ordenador es, sin embargo, de los de "28815 *bytes free*", entonces sí podéis efectuarla.

La aplicación consiste en almacenar en la *RAM* los datos de la memoria de vídeo que queramos. Tomemos por ejemplo el caso del modo *SCREEN 2,* cuyos datos son muy difíciles de trasladar debido a la escasa memoria libre que posee este modo de pantalla.

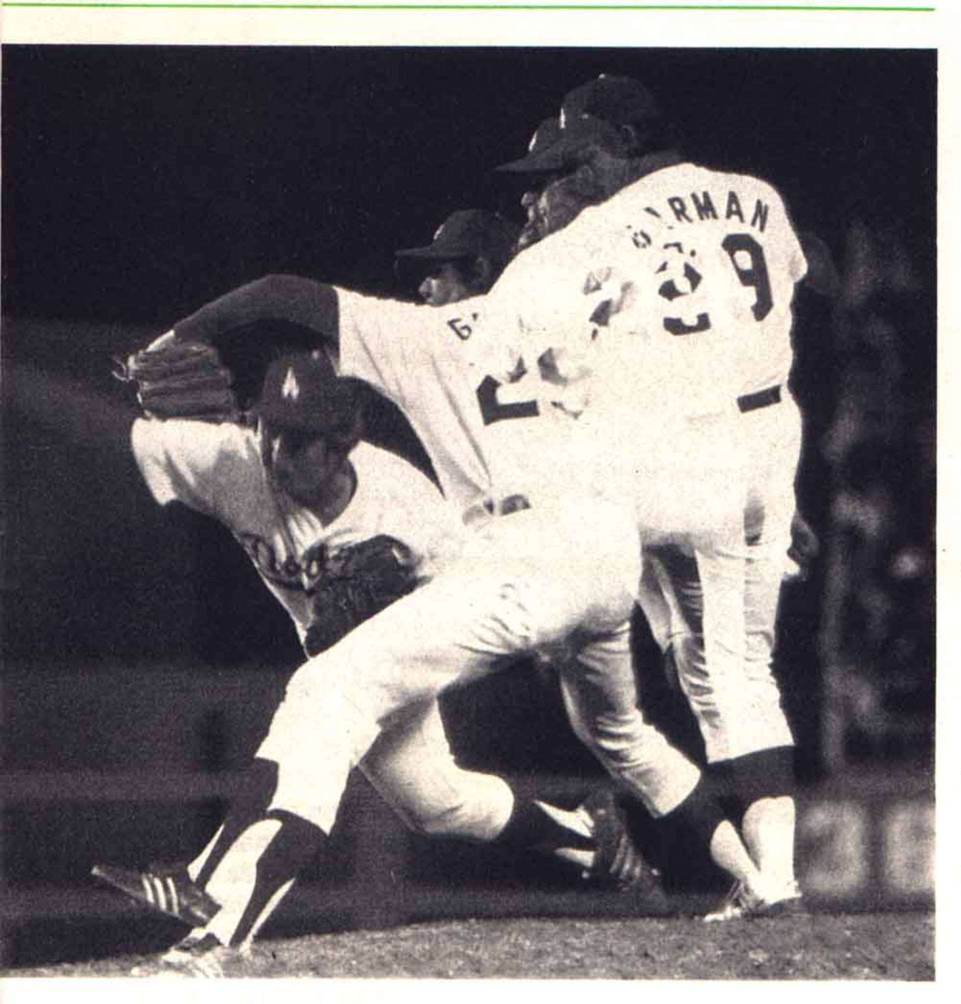
Imaginaos que en el juego que estáis programando se define un dibujo de buena calidad en el modo *SCREEN 2,* y que en medio del juego queréis pasar a *SCREEN 0* para escribir en la pantalla un mensaje con texto, y después queréis volver al mismo dibujo en el modo *SCREEN 2.* Sería demasiado largo tener que volver a dibujarlo, sobre todo si se utilizan para su diseño muchas funciones gráficas, como *CIRCLE, LINE, DRAW,* etc... Sin embargo, si trasladáis de lo que sea al observador con la pantalla en el modo *SCREEN 0,* y a continuación pasáis a *SCREEN 2* y tomáis de nuevo en la memoria de vídeo los datos que habíais dejado en la *RAM,* volveréis al mismo dibujo de la pantalla que teníais previamente.

Lo principalmente de este movimiento de bloques de datos entre las memorias es saber de qué parte de la memoria de vídeo a qué parte de la *RAM* se trasladan estos datos, y lo mismo se nos plantea en sentido inverso. En este caso vamos a trasladar todos los datos de la memoria de vídeo, los 16384



octetos, que es lo máximo que podéis trasladar, a la memoria *RAM.* Como ya sabréis, la memoria principal tiene 65536 octetos, numerados del 0 al 65535. El bloque del

0 32767 es la *ROM,* la memoria inalterable destinada a la lectura, y a partir del 32768 comienza la *RAM.* A partir de este octeto se almacena nuestro programa en *BASIC* (únicamente en el caso al que nos referimos, es decir, si vuestro ordenador MSX es de los de "28815 *bytes free*") y a partir de ahí podéis tomar cualquier bloque de octetos del final de la *RAM,* pues los últimos tres o cuatro mil octetos están destinados a otras funciones, cuyo nombre general es «sistema operativo». Tampoco os conviene tomar octetos de muy al principio, pues destruiríais total o parcialmente vuestro programa en *BASIC.* Si el bloque de datos a trasladar es muy grande, como en este caso (16384 octetos es un gran bloque) es necesario reservar una gran cantidad de memoria (otros 16384 octetos) en la *RAM.* Si empleáis los octetos que hay desde el 44000 al 60383, podréis almacenar vuestras pantallas, al tiempo que dejáis más de once mil octetos (desde el 32768 al 43999) para vuestros programas en *BASIC.* Este traslado de datos puede hacerse de una forma muy simple con un bucle *FOR-NEXT* para almacenar los datos, y otro bucle para retornarlos. (Te recordamos que en la memoria principal las órdenes *VPOKE* y *VPEEK* no se pueden utilizar, se utilizan las órdenes *POKE* y *VPEEK*). Pero los dos bucles, tanto el almacenamiento como el retorno de los datos en *BASIC* es verdaderamente lento, y sólo merece la pena emplear esta aplicación en el caso de que la pantalla que vayáis a almacenar sea muy complicada, y su trazado desde *BASIC* exija muchas órdenes gráficas.



Si sabéis Código Máquina, el traslado de los bloques de datos acudiendo al *BIOS* (Basic Input/Output System = Sistema Básico de Entrada/Salida) con nemónico *CALL* dirigido al código 0059H para trasladar datos de la memoria de vídeo a la *RAM* y el código 005CH para la operación inversa, es decir, trasladar datos de la *RAM* a la memoria de vídeo.

Si no sabéis Código Máquina no os preocupéis, pues analizaremos más profundamente en próximos números cómo desenvolverse en la memoria de vídeo desde el Código Máquina.

Tras explicaros el funcionamiento de todas las tablas y los procesos de transición entre los modos de pantalla, ya tenéis conocimientos suficientes como para aplicar en vuestros programas todos los trucos que se os ocurran basados en esta parte tan importante de vuestro ordenador.

Pero podéis aprender más de la memoria de vídeo de indudable utilidad: en el próximo número nos ocuparemos de los registros *VDP* y de sus aplicaciones para el programador.

**José M. Cavanillas**




# SON
# HB-5

Los ordenadores de la segunda generación están empezando su andadura en el mercado español. La competencia es dura y diversa. En este aspecto nos encontramos con el *Sony HB-500P,* que viene a competir directamente con el *Philips VG-8235.*

Varias son las características que hacen resaltar a este modelo del resto de los ordenadores de la gama. Su presentación unida a una solidez, a la que estamos poco acostumbrados, lo resaltan bastante. Sin embargo, existen cualidades dignas a tener en cuenta, entra otras, su teclado independiente.

Este, con un conjunto de teclas numéricas separado del teclado principal, le dan aspecto más semi profesional, lo que posibilita la entrada de datos numéricos con mayor rapidez. El tacto es muy bueno y la disposición de las teclas resulta ideal, al igual que el ángulo de inclinación del teclado, que se consigue gracias a dos pequeñas patas plegables.

La conexión al cuerpo principal del ordenador se hace a través de un cable de algo más de un metro. Esto es importante, ya que se puede tener el ordenador a una distancia cómoda, lo que ahorra espacio sobre todo en mesas con mucho trabajo.

Por otro lado, tenemos lo que es el ordenador es sí, donde podemos resaltar la unidad de discos de 3.5 pulgadas capaz de formatear a simple y doble cara, lo que permite acceder a 360 o 720K de información. Sin lugar a dudas, es-

ta cualidad, unida a que posee disco *RAM* de 32K, 256 colores de una paleta de 512, reloj, superposición de vídeo y todas esas ca-

# Y
# 500P

racterísticas de los ordenadores de la 2.ª Generación, le confieren en el modelo más completo del mercado. Todo esto se encuentra a buen recaudo, dentro de una carcasa similar a la del *IBM PC,* es más, cuando se ve por vez primera, da la sensación de estar delante del hermano pequeño de un *PC.* Aquí encontraremos todos los conectores que posee el estándar. Dos *port* de cartuchos y la unidad de disco, se encuentran en la parte anterior de la carcasa muy accesible al usuario, que agradecerá esta posibilidad. Además de esto, encontrará dos *ports* para *joysticks* y un *reset,* todo a mano. En la parte posterior de la carcasa, encontrará el *interface Centronics,* conectores para monitor *RGB,* audio/vídeo y para el cassette. También es de destacar un tercer bus de expansión para cartuchos *ROM* y un conector adicional para otra unidad de discos.

Aún queda por comprobar una cualidad común a todos los ordenadores MSX: la compatibilidad. De hecho, a este apartado habría que dedicarle algo más de tinta y tiempo, que no hemos podido en esta ocasión.

En suma, es un conjunto muy completo, al que no le falta de nada. Quizás sea demasiado para el usuario medio, pero lo que está claro es que el equipo se presenta como la más completa versión de un ordenador de la 2.ª Generación. Sin embargo, no todo va a ser bueno. El precio es su principal enemigo, puesto que al rondar una cifra de 140.000 ptas. no todos van a estar dispuestos a desembolsarla. Insistimos en que el equipo es completo, los manuales son buenos y que, además viene con un disco del sistema operativo *MSX-DOS.* Pero aún así, creemos que es demasiado aquilatado, teniendo en cuenta que se trata de una máquina con tecnología antigua (se mantiene el *Z-80*), aunque aprovechada al máximo.

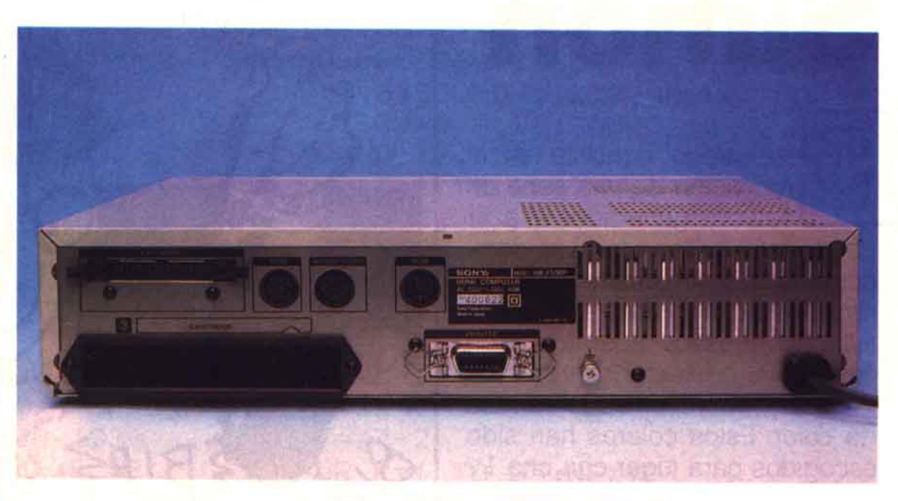



**Foto 1: Vista posterior del ordenador, donde podemos apreciar el bus de expansión, conector para unidad de disco y ports vídeo y cassette.**




**Foto 2: El teclado independiente es una de las grandes características de este potente modelo.**

# Simón

Este juego consiste en la repetición de una serie de sonidos generados al azar por nuestro ordenador. Los sonidos posibles son cuatro, con una representación gráfica en la pantalla de nuestros aparatos. Con cada nota, un sector diferente de los cuatro representados cambia de color. Estos colores han sido escogidos para jugar con una TV en color; si se juega en un aparato monocromático posiblemente habrá que modificar algún valor de la tabla de colores. Los colores están contenidos en las matrices unidimensionales c1(n), correspondiente a los colores sin sonido, y en la tabla c2(n), contiene los colores correspondientes a los sonidos. El elemento 1 corresponde al *sprite* 2 y 3, el elemento 2 al *sprite* 6 y 7, el elemento 3 al *sprite* 4 y 5, y el elemento 4 al *sprite* 0 y 1.




Las variables más importantes son las matrices de este programa; aparte de las matrices de colores, las otras matrices son:

– A(n), contiene la secuencia de sonidos a repetir. Inicialmente tiene 100 elementos, pero no hay inconveniente para aumentar la dimensión de esta matriz.

– NM$(n), contiene el nombre de los que han conseguido un récord.

– RE(n), contiene las puntuaciones récord.

El programa en sí no contiene ninguna gran complejidad estructural, y cada rutina tiene un *REM* que indica su función. Una posible mejora de este juego consiste en la optimización de su velocidad, para conseguir un desarrollo del juego más veloz, aunque considero que la dificultad en esta versión es suficiente.

El programa emplea las teclas de cursor, pero si se desea emplear un *joystick* no hay más que sustituir el 0 de la línea 260 por el número 1 ó 2, según donde esté, conectado el mando de juegos.

Por último, añadir que el programa consta de un temporizador que ocasiona que sólo se dispongan de 10 segundos para pulsar la tecla correspondiente al sonido en cada ocasión. Si se sobrepasa ese tiempo, se habrá perdido la partida.

**Diego Soriano**
**Madrid**

```
10 CLS
30 KEY OFF
40 CLEAR 1000
50 RA=RND(-TIME)
60 DIM A(100)
70 C1(1)=6:C1(2)=14:C1(3)=4:C1(
     4)=12
80 C2(1)=9:C2(2)=11:C2(3)=5:C2(
     4)=3
90 FOR P=1 TO 5:RE(P)=240-P*40:
     NEXT P
100  GOSUB 450
110 REM *** LAZO PRINCIPAL ****
120 FOR N=1 TO 100
130 A(N)=INT(RND(4)*4)+1
140 NEXT N
150 L=1
160 FOR P=1 TO L
```

```
170 A$=CHR$(67+A(P))
180 C(A(P))=C2(A(P)):GOSUB 1380
    :FOR T=1 TO 200:NEXT T:C(A
    (P))=C1(A(P)):GOSUB 1380
190 PLAY "V6L4T25004S11M12000XA
    $;"
200 FOR T=1 TO 100*DI:NEXT T
210 NEXT P
220 PRINT
230 LOCATE 8,21:PRINT"TU TURNO"
240 FOR P=1 TO L
250 TIME=0
260 IF STICK(0)=0 THEN Z=0  ELS
    E GOTO 260
270 IF TIME=500 THEN GOSUB 1230
280 IF Z=0 THEN Z=STICK(0) ELSE
     GOTO 280
290 IF Z=0 THEN 270
300 A2=INT(Z/2)+1
310 A2$=CHR$(67+A2)
320 PLAY "V6L4t25004S11M12000XA
    2$;"
330 C(A2)=C2(A2):GOSUB 1380:FOR
     T=1 TO 150:NEXT T:C(A2)=C
    1(A2):GOSUB 1380
340 IF A2<>A(P) THEN GOTO 430
350 PU=PU+1:LOCATE 8,19:PRINT"P
    UNTUACION";:PRINT USING "#
    ##";PU;
360 IF INT(PU/15)=PU/15 AND DI>
    .5THEN DI=DI-.25
370 NEXT P
380 PRINT
390 LOCATE 8,21:PRINT" ESCUCHA
    "
400 FOR T=1 TO 150:NEXT T
410 L=L+1
420 GOTO 160
430 GOSUB 1230
440 END
450 ' ***** PRESENTACION ******
    ***
460 COLOR 10,4,12
470 SCREEN 2,3
480 GOSUB 1130
490 X0=30:Y0=50
500 X1=30:Y1=82
510 X2=108:Y2=0
520 X3=130:Y3=0
530 X4=108:Y4=132
540 X5=130:Y5=132
550 X6=208:Y6=50
560 X7=208:Y.7=82
570 GOSUB 1860::SOUND 6,5:SOUND
     7,6:SOUND 1,120:SOUND 8,1
    6:SOUND 9,16:SOUND 10,16:S
    OUND 12,120:SOUND 13,4
580 FOR T=1 TO 30
590 X0=X0+2
600 X1=X1+2
610 Y2=Y2+1
620 Y3=Y3+1
630 Y4=Y4-1
640 Y5=Y5-1
650 X6=X6-2
660 X7=X7-2
670 FOR P=1 TO 35:NEXTP
680 PUT SPRITE 0,(X0,Y0),9
690 PUT SPRITE 1,(X1,Y1),9
700 PUT SPRITE 2,(X2,Y2),11
710 PUT SPRITE 3,(X3,Y3),11
720 PUT SPRITE 4,(X4,Y4),5
730 PUT SPRITE 5,(X5,Y5),5
740 PUT SPRITE 6,(X6,Y6),3
750 PUT SPRITE 7,(X7,Y7),3
760 NEXT T
770 FOR I=1 TO 4:C(I)=15:NEXT I
    :GOSUB 1380:SOUND 6,13:SOU
    ND 1,200:SOUND 3,150:SOUND
     7,0:SOUND 8,16:SOUND 9,16
    :SOUND 10,16:SOUND 12,30:S
    OUND 13,9:
780 C(4)=9:C(1)=11:C(3)=5:C(2)=
    3:FOR I=1 TO 30:NEXT I:GOS
    UB 1380:FOR T=1 TO 4:C(T)=
    C1(T):NEXT T:FOR I= 1 TO 1
    00:NEXT I:GOSUB 1860
790 GOSUB 1800
800 DRAW "S9BM30,150R16G4L5F9D2
    L16E4R5H9U2"
810 DRAW "BM+19,+0R12G4D7F4L12E
    4U7H4"
820 DRAW "BM+15,+0R4F5E5R4D15H4
    U6G5H5D6G4U15"
830 DRAW "BM+25,+0R9F4D7G4L9H4U
    7E4":DRAW "BM+3,+3R3F4D1G4
    L3H4U1E4":DRAW "BM+13,-3R4
    F8U5E4D15L4H8D5G4U15"
840 GOSUB 1800:PLAY "O4L4F","O2
    L4F","O6L4F"
850 FOR T=1 TO 2700: NEXT T
```

```
860 SCREEN 0:COLOR 1,3:WIDTH 35
870 PRINT"           - INSTRUCCI
    ONES -":PRINT:PRINT "   ES
    TE JUEGO CONSISTE EN REPET
    IR  LA SECUENCIA DE COLOR
    ES Y SONIDOS  PRODUCIDOS P
    OR LA MAQUINA."
880 PRINT "   PARA INTRODUCIRLO
    S SE UTILIZAN   LAS TECLAS
     DE CURSOR."
890 PRINT "   SI LA SECUENCIA E
    S REPETIDA        INCORRECTA
    MENTE,SE TERMINA LA       P
    ARTIDA."
900 PRINT"   EL GRADO DE DIFICU
    LTAD DETERMINA EL TIEMPO E
    NTRE CADA NOTA."
910 LOCATE 6,17:PRINT"¿GRADO DE
     DIFICULTAD [1 a 3]?":LOCA
    TE 7,18:PRINT "(1-rapido
    3-lento)"
920 DI$=INKEY$
930 IF DI$="" THEN 920
940 DI=VAL(DI$):IF DI>3 OR DI<1
     THEN 920
950 D1=DI
960 FOR T=1 TO 4:C(T)=C1(T):NEX
    T T:GOSUB 1350
970 RETURN
980 DATA 0,1,3,7,15,15,31,31,63
    ,63,127,127,127,255,255,25
    5,128,192,224,240,248,252,
    254,255,254,252,248,240,24
    0,240,224,224
990 '          ESPRITE      1
1000 DATA 255,255,255,127,127,1
    27,63,63,31,81,15,15,7,3,1
    ,0,224,224,240,240,240,248
    ,248,254,255,254,252,248,2
    40,224,192,128
1010 '       SPRITE 2
1020 DATA 0,0,0,3,15,31,63,127,
    255,127,63,31,15,7,3,1,7,6
    3,255,255,255,255,255,255,
    255,255,255,252,224,192,12
    8,0
1030 '           SPRITE 3
1040 DATA 224,252,255,255,255,2
    55,255,255,255,255,255,63,
    7,3,1,0,0,0,0,192,240,248,
    252,255,255,254,252,248,24
    0,224,192,128
1050 '           SPRITE 4
1060 DATA 1,3,7,15,31,63,127,25
    5,127,63,31,15,3,0,0,0,0,1
    28,192,224,252,255,255,255
    ,255,255,255,255,255,255,6
    3,7
1070 '               SPRITE 5
1080 DATA 0,1,3,7,63,255,255,25
    5,255,255,255,255,255,255,
    252,224,128,192,224,240,24
    8,252,254,255,254,252,248,
    240,192,0,0,0
1090 '            SPRITE 6
1100 DATA 1,3,7,15,31,63,127,25
    5,127,63,31,15,15,15,7,7,0
    ,128,192,224,240,240,248,2
    48,252,252,254,254,254,255
    ,255,255
1110 '            SPRITE 7
1120 DATA 7,7,15,15,15,31,63,12
    7,255,127,63,31,15,7,3,1,2
    55,255,255,254,254,254,252
    ,252,248,248,240,240,224,1
    92,128,0
1130 FOR P=0 TO 7:P$="":FOR T=1
     TO 32
1140 REM *** TRAT. SPRITES ***
1150 READ SP
1160 P$=P$+CHR$(SP)
1170 NEXT T
1180 SPRITE$(P)=P$
1190 PRINT
1200 NEXT P
1210 RESTORE
1220 RETURN
1230 CLS:PUT SPRITE 0,(0,208):L
    OCATE 8,19:PRINT " HAS FAL
    LADO       "
1240 REM *** SI SE FALLA: ***
1250 PRINT "TU PUNTUACION ES";P
    U;" PUNTOS"
1260 GOSUB 1480
1270 FOR T=1 TO 1700:NEXT T
1280 COLOR 10,1,4
1290 PRINT" JUEGAS OTRA VEZ (S/
    N)";
1300 J$=INPUT$(1)
1310 IF J$="N" OR J$="n" THEN C
    LS:PUT SPRITE 0,(0,208):LO
    CATE 7,10:PRINT:PRINT"¡ HA
    STA OTRA !":END
1320 IF J$="S" OR J$="s" THEN C
    LS:GOSUB 1710:TIME=0:DI=D1
    :PU=0:GOTO 120
1330 GOTO 1300
```

```
1340 REM *** CAMBIO COLOR SPRIT
     ES **
1350 COLOR 10,1,4
1360 SCREEN 1,3
1370 GOSUB 1130:GOSUB 1710
1380 PUT SPRITE 0,(87,50),C(4),
     0
1390 PUT SPRITE 1,(87,82),C(4),
     1
1400 PUT SPRITE 2,(105,30),C(1)
     ,2
1410 PUT SPRITE 3,(127,30),C(1)
     ,3
1420 PUT SPRITE 4,(105,102),C(3
     ),4
1430 PUT SPRITE 5,(127,102),C(3
     ),5
1440 PUT SPRITE 6,(145,50),C(2)
     ,6
1450 PUT SPRITE 7,(145,82),C(2)
     ,7
1460 RETURN
1470 REM *** RECORDS ***
1480 BA=0:FOR T=5 TO 1 STEP -1
1490 IF PU>RE(T) THEN BA=T
1500 NEXT T
1510 IF BA=0  THEN GOSUB 1820:R
     ETURN
1520 PUT SPRITE 0,(40,208):LOCA
     TE 8,21:PRINT STRING$(10,3
     2)
1530 LOCATE 3,5:PRINT"¡HAS BATI
     DO UN RECORD!":GOSUB 1840:
     FOR P=1 TO 15:COLOR 15,1,P
     :FOR T=1 TO 200:NEXT T:NEX
     T P
1540 COLOR 2,1,10
1550 LOCATE 6,7:DEF USR=342:JK=
     USR(2):LINE INPUT"¿NOMBRE?
     ";NM$:CLS
1560 NM$(BA)=NM$:RE(BA)=PU
1570 PRINT:PRINT "☺☺☺☺☺☺☺☺☺☺☺☺☺
     ☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺"
1580 FOR P=1 TO 19
1590 IF P/2=INT(P/2) THEN A$="
     B" ELSE A$="☺"
1600 LOCATE 0,P:PRINT A$:LOCATE
     31,P:PRINT A$;
1610 NEXT P
1620 PRINT "☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻☻
     ☻☻☻☻☻☻☻☻"
1630 LOCATE 10,3:PRINT"RECORDS"
1640 LOCATE 10,4:PRINT "▼▲▼▲▼▲▼
     "
1650 FOR P=1 TO 5
1660 LOCATE 6,5+2*P:PRINT NM$(P
     ):LOCATE 20,5+2*P:PRINT US
     ING"####";RE(P)
1670 NEXT P
1680 LOCATE 3,18
1690 RETURN
1700 REM *** TABLERO ***
1710 LOCATE 8,3:PRINT "┌────
     ──────────┐"
1720 FOR P=1 TO 13
1730 LOCATE 8,3+P:PRINT "│╲╱╱╲

     ╱╲╱╲╱╲╱│"
1740 NEXT P
1750 LOCATE 8,17:PRINT"└────
     ──────────┘"
1760 LOCATE 9,4:PRINT "*":LOCA
     TE 19,4:PRINT "*":LOCATE
     9,16:PRINT "*":LOCATE 19,
     16:PRINT "*"
1770 RETURN
1780 END
1790 REM *** MUSIQUILLA ***
1800 PLAY "V10S0M14000T250L4F05
     C#C04G#05FC#C04L8G#F..L4D#
     05C04A#G05D#C04A#L8GD#..",
     "V15S0M14000T250L402F03C#C
     02G#03FC#C02L8G#F..L4D#03C
     02A#G03D#C02A#L8GD#..","V1
     2S0M14000T250L406F07C#C06G
     #07FC#C06L8G#F..L4D#07C06A
     #G07D#C06A#L8GD#.."
1810 RETURN
1820 PLAY "V1304T110L4GL8ABB05C
     D.L16CL804B05DC04BAL4GT100
     L804DC03BAL4G."
1830 RETURN
1840 PLAY "V15T8006L6FL24GAGFEL
     10DL607DL16C06AA#AGFL8GR8L
     6FL24GAGFEL10D07L6DL16C06A
     A#AGFF","V14T8204L6FL24GAG
     FEL10DL605DL16C04AA#AGFL8G
     R8L604FL24GAGFEL10D05L6DL1
     6C04AA#AGFF"
1850 RETURN
1860 FOR I=1 TO 13:SOUND I,0:NE
     XT I:SOUND 7,184
1870 RETURN
```


GANE
7.000 PTAS.
todos los meses
BANCO DE ESPAÑA
2E1554046
MIL pesetas
1000
PARTICIPANDO EN NUESTRO CONCURSO
MSX Magazine premiará cada mes los programas que nos hagan llegar nuestros lectores.
Para participar en este concurso abierto, todo aficionado a los ordenadores con este estándar deberá hacer llegar a la redacción de la revista el listado, un cassette y un texto explicativo.
Entre todos los programas que recibamos cada mes, serán seleccionados para su publicación aquellos que reúnan los siguientes criterios:
— Originalidad de la aplicación.
— Simplicidad del método de programación.
La única condición para participar en el concurso será que los programas no hayan sido publicados previamente en ninguna revista.
Enviar vuestros programas a: MSX Magazine
C/Bravo Murillo, 377 - 5.º A 28020 MADRID
CINCO MIL pesetas
5000
14457150

# Diagramas de flujo

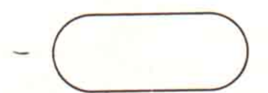
***A medida que vas avanzando en el conocimiento de BASIC, tus programas se van haciendo cada vez más complejos; llenos de instrucciones, sentencias y bifurcaciones. Por ello se hace muy útil el uso de DIAGRAMAS DE FLUJO (también llamados ORGANIGRAMAS).***

Un diagrama de flujo no es más que un boceto de la estructura del programa que nos sirve tanto para prepararlo como para recordar al cabo del tiempo cómo ha sido realizado.

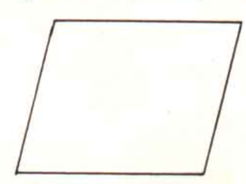
Personalmente, creo que no es necesario realizar los programas de flujo con excesivo detalle, basta con que una simple ojeada al diagrama permita conocer las ideas principales del programa.

Para hacer más comprensible y sencillo un diagrama de flujo utilizamos CAJAS con significados particulares, y dentro de ellas indicamos las variables implícitas, y en su caso, las operaciones a realizar.

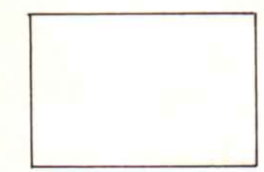
De momento vamos a conocer sólo algunas de esas cajas, que nos permitirán desarrollar nuestros próximos programas. A medida que vayamos avanzando, iremos introduciendo las cajas que necesitemos.



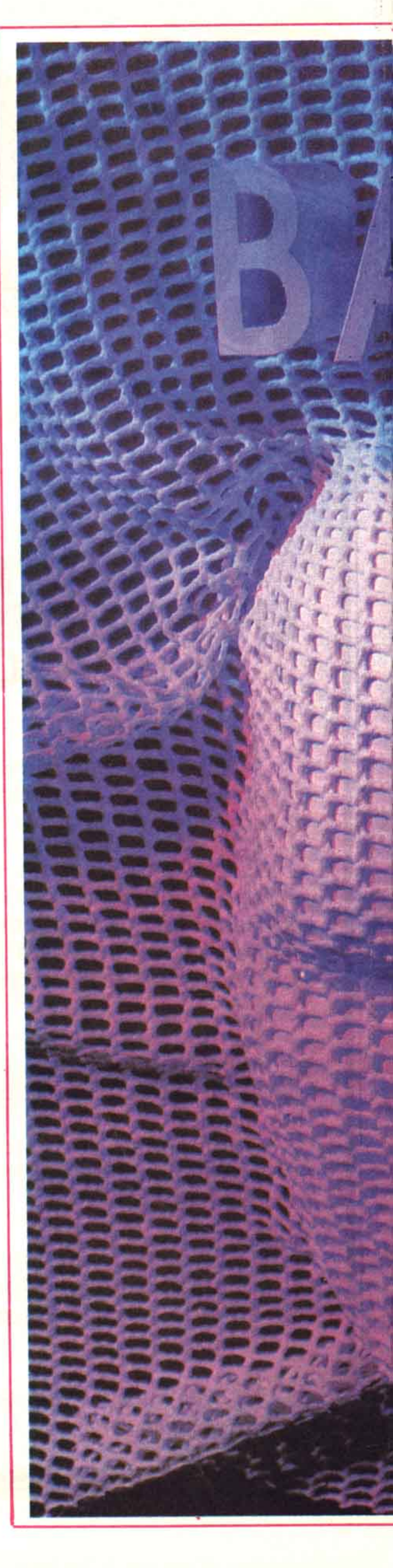
Se utiliza para indicar el comienzo o final de un programa. Dentro pondremos E (Empezar) o FIN según sea principio o final.



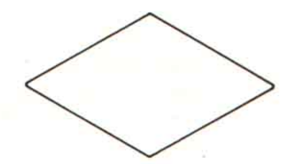
Sirve para indicar que se han de introducir datos en el ordenador. Dentro de este símbolo indicaremos los nombres de las variables.



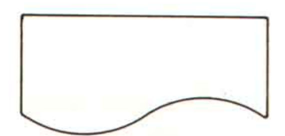
El rectángulo indica que se va a realizar una operación o cálculo.



Dentro de esta caja se indica la operación u operaciones a realizar.



El rombo significa toma de decisión. Según sea cierta o no la comparación que se ha escrito en su interior, el programa continuará por uno y otro vértice del mismo.



Significa presentar un resultado. Lo utilizaremos indistintamente para que ese resultado se presente por pantalla, impresora, cinta magnética o disco.

Recordemos el programa del artículo del mes pasado; con él calculábamos las raíces de una ecuación de segundo grado. En la figura 6(a) tienes el diagrama de flujo correspondiente. El diagrama de la figura 6(b) es algo más simple y menos detallado que el de la figura 6(a).

Existen otros símbolos para expresar las mismas operaciones, pero, tratándose de un elemento auxiliar, creo que lo más útil es que utilices el menor número posible de ellos. Lo importante de un diagrama de flujo es que puedas reconocerlo (y puedan otros reconocerlo) al cabo de algún tiempo.

## FOR...TO...STEP...NEXT

Supongamos que queremos calcular el cuadrado, el cubo, la raíz cuadrada y la raíz cúbica de los números comprendidos entre 1 y N. Con los conocimientos que

E

DIMENSIONAR
NB$ (100), DI$ (100),
TE! (100)

I = 0

I = I+1

NOMBRE(NB$(I))
DIRECCION(DI$(I))
TELEFONO(TE!(I))

¿OTRO NUMERO?
A$

A$ = "S"

SI

NO

NOMBRE(NB$)

NB$(I1) = NB$

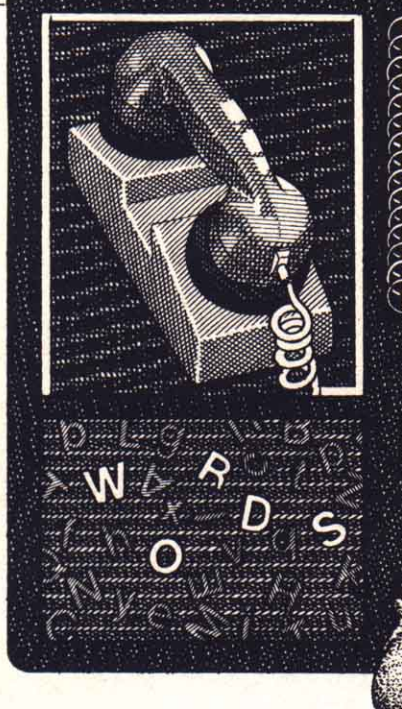
SI

NOMBRE:(NB$(I1))
DIRECCION:(DI$(I1))
TELEFONO:(TE!(I1))

NO

I1 = I,1

**Diagrama 1**



cuando trabajemos con variables con subíndice se lo indiquemos, y además que le digamos el tamaño de dicha variable, es decir, cuál es el valor máximo que va a tomar el subíndice. Esto lo hacemos por medio de la sentencia *DIM* (DIMENSION) de la siguiente forma:

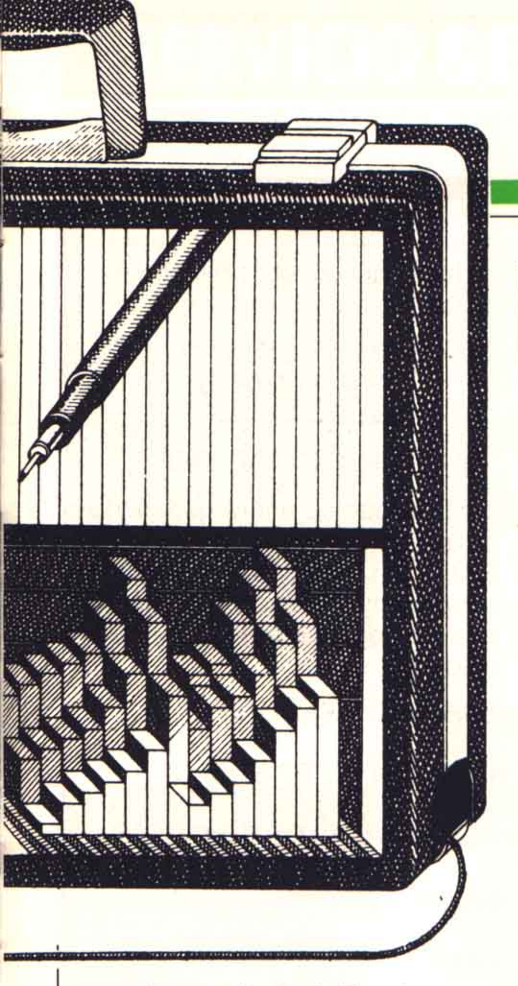
*DIM NB$*(100)

De esta forma le decimos al ordenador que la variable *NB* es de caracteres y que va a contener 100 variables. Es decir, en nuestro ejemplo vamos a hacer un listín telefónico en que utilizaremos la variable alfanumérica con subíndice *NB* que contendrá el nombre de hasta 100 personas.

Como vamos a almacenar en el ordenador, además del nombre, la dirección y el número de teléfono definiremos otras dos variables, que llamaremos *DI* y *TE*, y por tanto al dimensionarlas escribiremos:

DIM NB$(100), DI$(100), TE!(100)

(Recuerda que un número ente-



ro no tiene más de 5 cifras. Por eso hemos hecho la variable *TE* de simple precisión).

Fíjate ahora en el programa 1 (cuyo diagrama de flujo es el diagrama 1). Conocemos ya casi todo lo que ahí se dice. Sin embargo, no conocemos la instrucción de la línea 40:

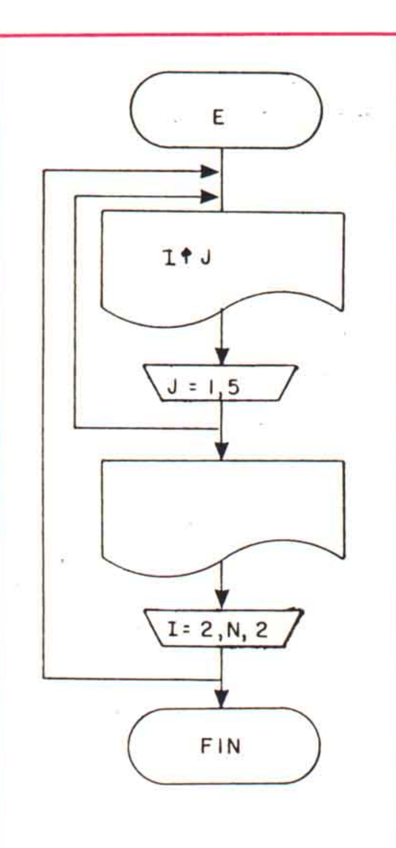
40 CLEAR 20000

Cuando conectamos el ordenador, éste reserva automáticamente 200 octetos para variables alfanuméricas. Por tanto, sólo podremos tomar ese tipo de variable con una longitud máxima de 200 caracteres. Si tratas de hacer, por ejemplo que la variable *A$* valga:

*A$*= «En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches duelos y quebrantos»,

E
DIMENSIONAR A$(100,2)
B$(0)="NOMBRE"
B$(1)="DIRECCION"
B$(2)="TELEFONO"
I = 0
I = I+1
A$ (I,I1)
I1 = 0,2
(¿OTRO NUMERO?) A$
A$ = "S"
SI
NO
(NOMBRE) NB$
NB$=A$(I1,0)
SI
NO
I1 = 1, I
B$ (I 2)
A$(I1,I2)
I2 = 0,2

**Diagrama 2**




```
10 INPUT N
20 FOR I=2 TO N STEP 2
30 FOR J=1 TO 5
40 PRINT I; "↑"; J; "="; I↑J
50 NEXT J
60 PRINT
70 NEXT I
80 END
```

**FIGURA 9b**

Si el valor P especificado en *STEP P* es 1, se puede omitir y escribir:

```
10 FOR I = A1 TO A2
.
.
.
.
100 NEXT I
```

A1, A2 y P pueden ser constantes o variables, y pueden ser enteros o reales, positivos o negativos. Así, se puede encribir:

```
10 FOR I=A TO A*A STEP
   SQR(A)
```

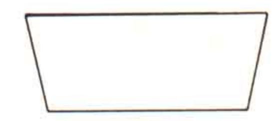
Sí, por ejemplo, escribimos:

```
10 FOR I = 1 TO −10 STEP 2
```

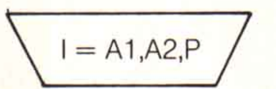
I empezará valiendo 1 y con este valor se realizarán los cálculos situados entre esta línea y *NEXT* I. El siguiente valor de I es I = 1 + 2 = 3. *BASIC* comprueba si este nuevo valor de I está entre 1 y −10 (entre A1 y A2) y, al no ser así, pasa a ejecutarla la línea siguiente a *NEXT* I.

Con esta nueva instrucción, nuestro programa quedará como se muestra en la figura 8(b), cuyo diagrama de flujo es el de la figura 8(a).

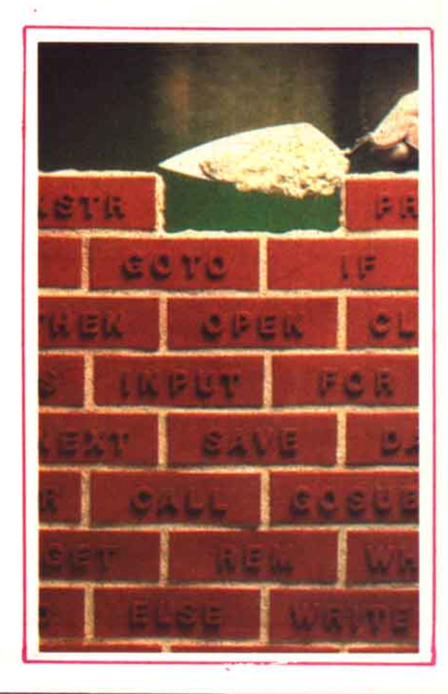
Evidentemente esto es más cómodo que lo que hacíamos antes.



Se utiliza para repetir un trozo de programa desde que cierta variable (I) tome un determinado valor hasta que alcance otro. Se escribe:




(FOR I = A1 TO AE STEP P)




Supongamos ahora que, dado N, queremos hallar las potencias 2, 3, 4 y 5 de los números pares comprendidos entre 1 y N. *BASIC* permite hacer bucles *FOR – NEXT* anidados, es decir:

```
FOR I=A1 TO A2 STEP P
...
FOR J=B1 TO B2 STEP Q
...
FOR K=C1 TO C2 STEP R
...
NEXT K
...
NEXT J
...
NEXT I
```

En *BASIC–MSX* se pueden anidar tantos bucles *FOR–NEXT* como se desee. Lo único que se exige es que no se solapen unos con otros. Es decir no es posible hacer lo siguiente:

```
FOR I=A1 TO A2 STEP P
...
FOR J=B1 TO B2 STEP Q
...
NEXT I
...
NEXT J
```

Si nos salimos de un bucle *FOR-NEXT,* los índices se inicializan. Es decir, en el caso anterior, cuando el ordenador se encuentra con *NEXT* I, toma para J el valor B1 (y no J + 1).

Por tanto podemos resolver el problema que nos hemos planteado con el programa de la figura 9(b), según el diagrama de la figura 9(a).

**J. Antonio Feberero**

# SERVICIO DE EJEMPLARES ATRASADOS

**ESTOS SON LOS EJEMPLARES DE MSX MAGAZINE APARECIDOS EN EL MERCADO CON UN RESUMEN DE SU CONTENIDO**




**Núm. 1**
¿Qué es el MSX? Su BASIC, periféricos, programas, software.




**Núm. 2**
Generación de sonido. MSX-DOS, el ordenador por dentro, programas, noticias.




**Núm. 3**
Los joysticks, 256 caracteres programables, Z80 corazón de león, compro/vendo/cambio.



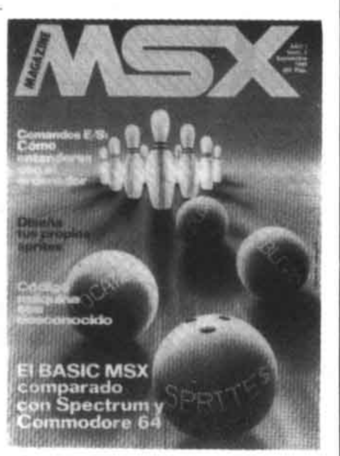

**Núm. 4**
Las comunicaciones entre ordenadores, la jerga informática, trucos, rincón del lector.




**Núm. 6**
Los 8 magníficos (test gigante), el bus de expansión, los misterios de la grabación, programas.




**Núm. 8**
Compact Disc, el periférico del futuro. Test: Dynadata DPC-200. Continuamos con la memoria de video. Libros, software, programas, trucos.




**Núm. 5**
Comandos de entrada/salida, el BASIC MSX comparado con Spectrum y Commodore 64. Código Máquina.




**Núm. 7**
Analizamos el Generador de Sonido. Aplicaciones matemáticas con el ordenador. La memoria de video. Trucos, noticias.




**Núm. 9**
Características técnicas del Compact Disc. Tratamiento de datos. Test: Quick Disk. Trucos, libros, noticias, programas.

**PARA HACER SU PEDIDO, RELLENE ESTE CUPON, HOY MISMO Y ENVIELO A MSX MAGAZINE BRAVO MURILLO, 377. Tel. 733 96 62 - 28020 MADRID**

Ruego me envien los siguientes números atrasados ..........................

al precio de **250** ptas. cada uno. Cuyo importe abonaré:

☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO

☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Número de mi tarjeta ..........................

Fecha de caducidad ..........................

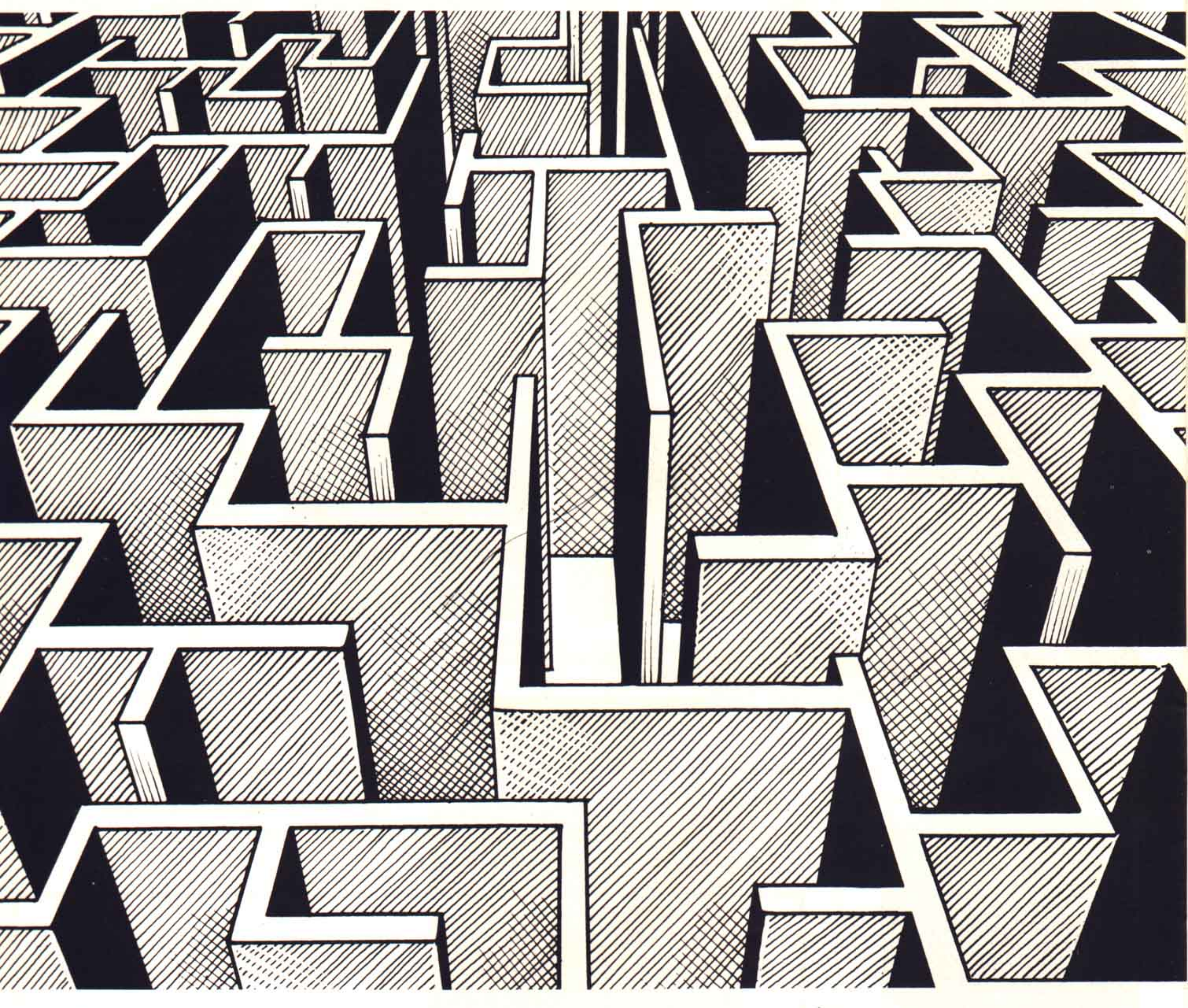
NOMBRE ..........................

DIRECCION ..........................

POBLACION .......................... C.P. ..........

PROVINCIA ..........................

# 3-D Bola



No hace mucho, aparecieron en los bares, unas maquínas con juegos cuyo objetivo era conducir una bola a lo largo de un laberinto en tres dimensiones. Este juego ha servido como inspiración a Alberto López, que nos ha enviado una entretenida versión de ese juego para los MSX.

Se trata de dirigir la bola por un laberinto en el que existen subidas, bajadas, ascensores, pinchos mortales y otros objetos, todo ello representado en tres dimensiones. Si se consigue llegar a la meta (que es un punto) antes de que se acabe el tiempo, pasaremos a la pantalla siguiente. El juego consta de tres pantallas diferentes.

```
10 COLOR 15,1,1
20 OPEN"GRP:"AS#1
30 SCREEN2,2
40 GOSUB870
50 SCREEN2
60 FOR G=0 TO 2:PUTSPRITE G,(0,0
   ),0,10
70 PLAY"S1M500007C64R1C64R1C64R1
   C64R1C64R1"
80 PLAY"V15S1M3000003C104","V15S
   1M3000006C104"
90 PT=1
100 GOSUB 1390
110 GOSUB 1120
120 B=6
130 GOSUB 500
140 X=X1:Y=Y1:TN=TM:GOSUB 1070
150 VX=0:VY=0
160 A=STICK(0) OR STICK(1)
170 VX=VX-(A=2 OR A=3 OR A=4)+(A
    =6 OR A=7 OR A=8)
180 VY=VY-(A4 OR A=5 OR A=6)+(A=
    8 OR A=1 OR A=2)
190 CR=VPEEK(6144+((Y+16)\8)*32+
    (X+8)\8)
200 ON CR GOTO 210,210,280,280,2
    80,280,280,320,320,360,450,
    730,210,560,210,1080
210 X=X+VX-VY
220 VX=VX\1:VY=VY\1
230 Y=Y+VY
240 PUTSPRITE 0,(X,Y),8,0
250 PUTSPRITE 1,(X,Y),15,1
260 TN=TN-.04:IF TN=TN\1 THEN GO
    TO 1030
270 GOTO160
280 VY=VY+.5
290 Y=Y+VY
300 X=X-(VY/2)+VX
310 GOTO 240
320 VX=VX+.5
330 X=X+VX-VY
340 Y=Y+VX+VY
350 GOTO240
360 PLAY"O5S8M1000C4R5C4","O7S8M
    1000C4R5C4"
370 FORG=1 TO 11
380 PUTSPRITE 0,(X,Y),8,2
390 FOR J=1 TO 40:NEXT
400 PUTSPRITE 0,(X,Y),8,0
410 FORJ=1 TO40:NEXT
420 NEXT
430 FORG=1 TO 500:NEXT
440 GOTO 450
450 PUTSPRITE 0,(X,Y),8,2
460 PLAY"V15S1M900007B104","V15S
    1M900005G104","V15S1M900008
    D104"
470 FORG=1 TO800:NEXT
480 GOSUB 500
490 GOTO 140
500 'VIDAS
510 B=B-1
520 FORG=6144 TO 6150:VPOKE G,10
    : NEXT
530 IF B=0 THEN 1630
540 FORG=6144 TO 6143+B:VPOKE G,
    13:NEXT
550 RETURN
560 'PASPANTALLA
570 Z=0:T=0
580 Z=Z+(X-8-Z)/10:T=T+(Y+4-T)/1
    0
590 PUTSPRITE 2,(Z,T),11,3
600 IF X-8-Z>2 OR Y+4-T>2THEN 58
    0
610 FORX=X TO X-8 STEP-1
620 FOR G=1 TO 20:NEXT
630 PUTSPRITE 0,(X,Y),8,0
640 PUTSPRITE 1,(X,Y),15,1
650 NEXT
660 PLAY"L3V1504CG05CL4","L3V150
    3CG04CL4","L3V1506CG07CL4"
670 FOR Y=Y TO-22 STEP-1
680 PUTSPRITE 1,(X,Y),15,1:PUTSP
    RITE 0,(X,Y),8,0:PUTSPRITE
    2,(X,Y+4),11,3:NEXT
690 PT=PT+1:IF PT=4 THEN PT=1
700 GOSUB1390
710 B=B+1
720 GOTO130
730 'SALTO
740 PLAY"S1M2500003C2"
750 PUTSPRITE 2,(((X+8)\8)*8,((Y
    +16)\8)*8-10),15,4
760 FOR VZ=-8 TO8 STEP.5
770 X=X+VX/2-VY/2
780 Y=Y+VY/2+VZ/2
```

```
790 IF X>240 OR X<-0 OR Y>180 TH
    EN 850
800 PUTSPRITE 0,(X,Y),8,0
810 PUTSPRITE 1,(X,Y),15,1
820 NEXT
830 CR=VPEEK(6144+((Y+17)\8)*32+
    (X+8)\8)
840 IF CR<10 THEN PLAY"S1M990006
    C204","S1M990005C04"
850 PUTSPRITE 2,(0,0),0,10
860 GOTO240
870 'SPRITES
880 RESTORE980
890 FOR G=0 TO 4
900 S$=""
910 FOR F=1 TO 32
920 READ A$
930 S$=S$+CHR$(VAL("&H"+A$))
940 NEXT
950 SPRITE$(G)=S$
960 NEXT
970 RETURN
980 DATA 07,1E,38,70,63,C7,CF,1F
    ,9F,9F,DF,7F,7F,3F,1F,07,E0
    ,38,1C,FE,FE,FF,FF,FF,FF,FF
    ,FF,FE,FE,FC,F8,E0
990 DATA 0,3,F,1F,3F,3F,7F,7F,7F
    ,7F,3F,3F,1F,F,3,0,0,C0,F0,
    F8,FC,FC,FE,FE,FE,FE,FC,FC,
    F8,F0,C0,0
1000 DATA 5,1d,3a,7B,77,87,EA,ED
    ,1D,FA,D7,2F,6E,31,1F,7,E0,
    F8,20,DE,DE,ED,33,F7,E3,DC,
    2F,76,76,74,78,A0
1010 DATA 0,0,0,0,0,0,0,0,0,7,F,
    1F,3F,7F,FF,0,0,0,0,0,0,0,0
    ,0,0,FF,FE,FC,F8,F0,E0,0
1020 DATA 31,4E,86,79,1,31,4E,86
    ,79,1,2,C,70,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0
1030 IF TN=-2THEN 360
1040 PLAY"S1M500006C32"
1050 VPOKE 6174-TN,10
1060 GOTO160
1070 FOR G=6175-TM TO 6175:VPOKE
     G,15:NEXT:RETURN
1080 VX=-2*VX
1090 VY=-2*VY
1100 PLAY"S1M2500003C2"
1110 GOTO210
1120 'CARACTERES
1130 RESTORE 1220
1140 FOR G=0 TO 135
1150 READ F$,C$
1160 F=VAL("&H"+F$)
1170 C=VAL("&H"+C$)
1180 VPOKE G,F:VPOKE G+2048,F:VP
     OKE G+4096,F
1190 VPOKE 8192+G,C:VPOKE 10240+
     G,C:VPOKE 12288+G,C
1200 NEXT
1210 RETURN
1220 DATA FF,51,2,51,4,51,8,51,1
     0,51,20,51,40,51,FF,51
1230 DATA FF,51,2,51,4,51,8,51,1
     0,51,20,51,40,51,80,51
1240 DATA 1,51,2,51,4,51,8,51,10
     ,51,20,51,40,51,FF,51
1250 DATA 10,71,10,71,20,71,20,7
     1,40,71,40,71,80,71,80,71
1260 DATA 1,71,1,71,2,71,2,71,4,
     71,4,71,8,71,0F,71
1270 DATA 10,71,10,71,20,71,20,7
     1,40,71,40,71,80,71,FF,71
1280 DATA 1,71,1,71,2,71,2,71,4,
     71,4,71,8,71,F8,71
1290 DATA 1,71,1,71,2,71,2,71,4,
     71,4,71,8,71,FF,71
1300 DATA1,51,2,51,4,51,8,51,10,
     51,20,51,40,51,80,51
1310 DATA 80,51,40,51,20,51,10,5
     1,8,51,4,51,2,51,1,51
1320 DATA 0,1,0,1,0,1,0,1,0,1,0,
     1,0,1,0,1
1330 DATA 18,F5,18,F1,3C,A1,7E,A
     1,7E,B1,3C,B1,0,11,FF,55
1340 DATA FF,51,0,11,3E,91,7E,91
     ,7C,91,0,11,0,11,FF,51
1350 DATA 7E,F1,C0,F8,80,F8,FF,8
     8,FF,88,FF,88,FF,88,7E,81
1360 DATA FF,51,0,11,0,11,C,91,0
     ,11,0,11,0,11,FF,51
1370 DATA 10,4F,18,4F,1C,4F,FE,4
     F,FE,47,1C,47,18,47,10,47
1380 DATA FF,51,00,11,3C,21,62,2
     1,5E,21,7E,21,00,11,FF,55
1390 'PANTALLA
```

# Catálogo de Software para ordenadores personales IBM




Todo el Software disponible en el mercado reunido en un catálogo de 800 fichas

1.ª ENTREGA
550 FICHAS
+ FICHERO

Resto en dos entregas trimestrales de 150 fichas cada una




## PRECIO TOTAL DE LA SUSCRIPCION 8.000 PTAS.

COPIE O RECORTE ESTE CUPON DE PEDIDO

CUPON DE PEDIDO

SOLICITE HOY MISMO EL CATALOGO DE SOFTWARE A:

**infodis, s.a.**

Bravo Murillo, 377, 5.° A
28020 MADRID

O EN CONCESIONARIOS IBM

El importe lo abonaré POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO ☐

Cargue **8.000** ptas. a mi tarjeta American Express ☐ Visa ☐ Interbank ☐

Número de mi tarjeta ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

NOMBRE ____________________

CALLE ____________________

CIUDAD ____________________ C. P. __________

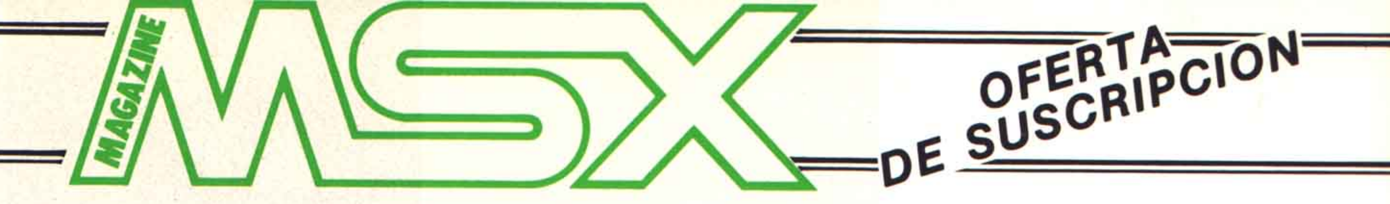
PROVINCIA ____________________ TELEFONO __________

ref: CATALOGO DE SOFTWARE

CS-2

```
1400 FOR G=6144 TO 6912:VPOKE G,
     10:NEXT
1410 ON PT GOTO 1600,1610,1620
1420 READ TM,X1,Y1,N
1430 GOSUB 1070
1440 FOR G=1 TO N
1450 READ S,C,L,AV
1460 CU=S
1470 FOR J=1 TO L
1480 VPOKE 6144+CU,C
1490 CU=CU+AV
1500 NEXT
1510 NEXT
1520 RETURN
1530 DATA 15,113,50,35,343,0,9,3
     1,344,1,8,31,404,2,7,31,339
     ,2,2,31,270,0,2,31,271,1,2,
     31,340,14,4,1,331,4,2,63,33
     2,6,2,63,363,5,2,63,364,3,2
     ,63,457,2,3,31,458,0,3,1,46
     1,1,2,62,490,0,18,1,459,16,
     2,62,374,11,2,115,520,0,2,2
     ,496,12,2,42
1540 DATA 497,1,1,1,498,10,1,1,4
     99,2,1,1,508,1,2,31,537,2,1
     ,1,269,2,2,31,340,0,3,1,371
     ,16,1,1,372,0,2,1,592,0,3,1
     ,529,12,2,69,595,1,1,1,597,
     2,2,31,599,0,2,1,601,1,2,31
     ,629,11,3,1
1550 DATA 20,12,150,49,328,2,2,3
     1,329,11,3,1,332,1,2,31,334
     ,2,2,31,335,16,2,1,337,12,2
     ,287,338,1,2,31,360,0,3,1,3
     66,0,3,1,390,4,4,63,391,6,4
     ,63,422,5,4,63,423,3,4,63,3
     93,4,2,63,394,6,2,63,425,5,
     2,63,426,3,2,63,462,2,2,57,
     466,1,2,57,520,0,3,1,396,4,
     2,63
1560 DATA 397,6,2,63,428,5,2,63,
     429,3,2,63,399,4,2,3,400,6,
     2,3,431,5,2,3,432,3,2,3,463
     ,0,3,1,496,4,2,63,497,6,2,6
     3,528,5,2,63,529,3,2,63,623
     ,0,1,1,371,2,2,294,372,0,2,
     1,374,1,2,200,375,9,6,33,37
     6,8,5,33,377,9,4,33,406,9,2
     ,33,407,8,2,33,506,8,2,33
1570 DATA 538,9,2,33,572,2,3,31,
     573,14,4,31,605,1,3,31,622,
     2,2,20,625,1,2,18
1580 DATA 15,117,28,39,174,2,2,3
     1,175,0,1,1,176,1,2,30,236,
     4,2,63,237,6,2,63,268,5,2,6
     3,269,3,2,63,360,2,4,31,361
     ,0,4,31,362,0,4,31,363,0,4,
     31,364,0,4,31,365,1,4,31,36
     4,11,1,1,392,11,2,62,394,11
     ,2,63,486,4,3,63,487,6,3,63
     ,518,5,3,63,519,3,3,63,675,
     2,2,31
1590 DATA 676,1,2,34,707,0,3,1,7
     10,12,1,1,653,0,3,31,654,0,
     3,31,656,1,3,31,655,11,2,62
     ,686,12,1,1,537,2,4,31,538,
     0,1,1,539,14,2,1,541,1,4,31
     ,652,2,3,31,569,0,3,1,600,0
     ,3,1,631,0,3,1,711,1,1,1,56
     9,16,2,1
1600 RESTORE 1530:GOTO1420
1610 RESTORE 1580:GOTO1420
1620 RESTORE 1550:GOTO1420
1630 'FIN
1640 SCREEN2
1650 PSET(52,80)
1660 DRAW"S40C7EEEEERRGLGRGLGGL"
1670 PSET (82,80)
1680 DRAW"C5EEERGGGL"
1690 PSET (122,40)
1700 DRAW"c5ERGL"
1710 PSET (102,80)
1720 DRAW"C4EEEEERDDDDEEEERGGGGG
     LUUUUGGGGL"
1730 PRESET(12,146)
1740 PRINT#1,"<SPACE> O BOTON DE
     L JOYSTICK"
1750 PRESET(44,160):PRINT#1,"PAR
     A VOLVER A JUGAR."
1760 LINE (0,177)-(255,177),15
1770 K=K+1:PUTSPRITE 0,(K,160),5
     ,0:PUTSPRITE 1,(K,160),15,1
1780 IF STRIG(0)=0 AND STRIG(1)=
     0 THEN 1770 ELSE 50
1790 FOR G=1 TO 1000:INPUT A$:PR

     INTHEX$(VAL("&B"+A$)):NEXT
```

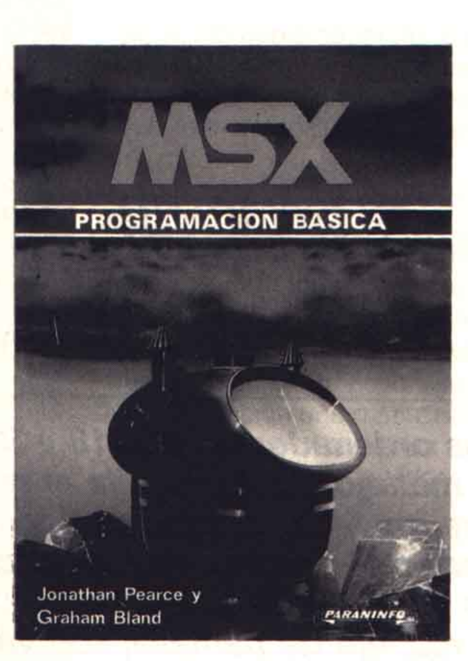



**LA REVISTA IMPRESCINDIBLE**
**PARA LOS USUARIOS DE LOS ORDENADORES PERSONALES MSX.**

Una publicación mensual que ayuda a obtener el máximo partido a su ordenador.

MSX publica cada mes programas y juegos, además de reportajes sobre programación y la posibilidad de ganar premios realizando programas y otros temas siempre de gran interés.

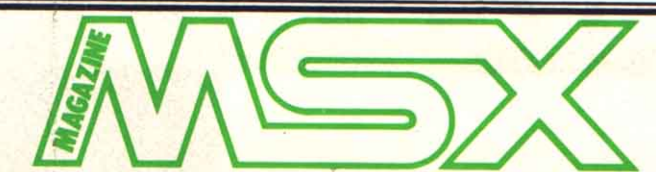



**ADEMAS,** beneficiese de un **15 %** **DE DESCUENTO** sobre el precio real de suscripción

| PRECIO NORMAL DE SUSCRIPCION | USTED SOLO PAGA | AHORRO |
|---|---|---|
| ~~3.600~~ PTAS. | 3.060 PTAS. | 15 % |

**APROVECHE AHORA** esta irrepetible oportunidad para suscribirse a **MSX**. Envie **HOY MISMO** la tarjeta adjunta a la revista, que no necesita sobre ni franqueo. Deposítela en el buzón más cercano. Inmediatamente recibirá su primer ejemplar de **MSX** más el **REGALO**. Y así durante un año (12 números).




Bravo Murillo, 377
Tel. 733 79 69
28020 MADRID

## El Procesador de vídeo

**(III)**

El procesador TMS 9918 A maneja la pantalla dividiéndola en 35 planos superpuestos numerados del 0 al 1 (Fig. 1).

Plano 34. Vídeo externo.
Plano 33. Color de fondo.
Plano 32. Diibujos y carácteres.
Planos 0 a 31. Sprites.

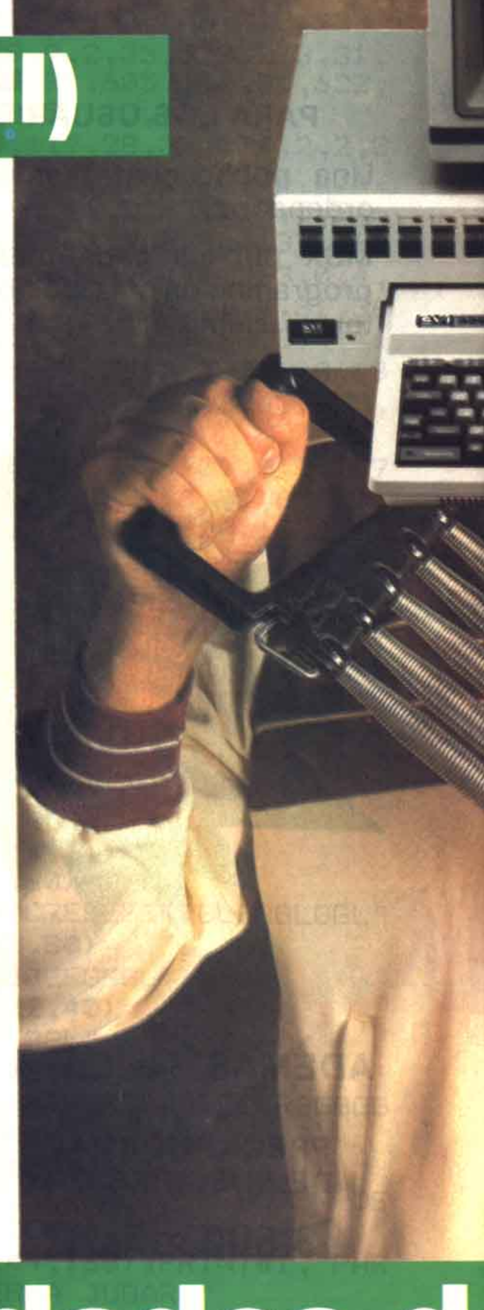
El plano más interno es el plano 34 y el más externo el cero. Esto quiere decir que las figuras de los planos con menor número de denominación taparán a las figuras de los planos con mayor numeración cuando coincidan con ellas.

El plano 34 está incluido en el vídeoprocesador para presentar en él figuras de una fuente de vídeo externa, como puede ser un aparato de vídeo cintas o también otro procesador de vídeo. Pero para nuestra desdicha, tal posibilidad no está habilitada en nuestros modelos de ordendor.

El plano 33 es un plano monocromático que proporciona el color de fondo en el modo de texto *(SCREEN 0)*, o el borde en el caso de las pantallas gráficas *(SCREEN 1 y SCREEN 2)*.

El plano 32 es el plano donde aparecen los carácteres gráficos

***Los ordenadores SVI-318 y SVI-328 utilizan para el manejo de la pantalla el mismo procesador que los MSX, el procesador TMS 9918 A. Por esto, muchos de los artículos publicados en esta revista en números anteriores referentes al manejo de pantalla, son válidos también para nuestro ordenador con algunas pocas diferencias.***



# Interioridades de SVI-318 y SVI-3




en el modo de texto, carácteres y dibujos en el modo de alta resolución *(SCREEN 1),* o la mezcla multicolor en el modo de gráficos de baja resolución *(SCREEN 2).*

Los restantes planos, del 31 al 0, son los planos por los que se moverán los *sprites.* Cada plano contendrá un solo *sprite,* y cuando dos de estos coincidan, cada uno en su plano, en las mismas coordenadas, quedarán superpuestos y se verá en primer término aquel que se mueva en un plano de orden menor.

## Los registros del procesador de vídeo

El procesador de vídeo, también conocido con el nombre de *VDP* (Vídeo Display Procesor), tiene a su disposición una memoria *RAM* de 16 *Kbytes* conocida como *VRAM* o memoria de vídeo.

La *VRAM* contiene datos que necesita el vídeo procesador para saber qué es lo que tiene que presentar en cada momento en la pantalla.

La *VRAM* está dividida en varias zonas, cada una de las cuales contiene datos con un determinado significado como veremos más adelante.

Para saber en cada momento qué zona de la *VRAM* tiene que consultar, así como la forma en que ha de tratar los datos de la misma, el *VDP* mira o consulta la información contenida en nueve registros internos, la figura 2 es una tabla que muestra qué es lo que significa dicho contenido.

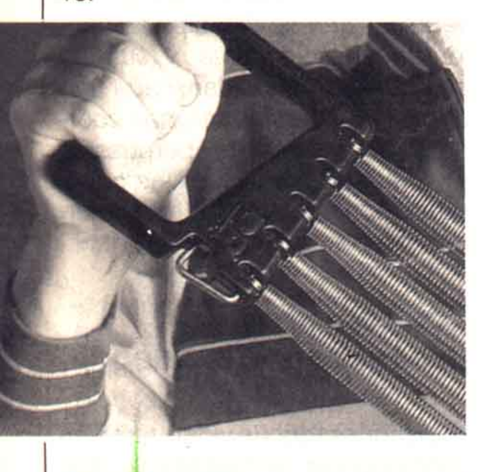
A contitnuación pasamos a explicar detenidamente lo que significa cada uno de ellos.

**Registro cero:**

- El *bit* cero de este registro habilita el plano 34, que era el plano re-

cepción de imágenes de vídeo externo y, por lo tanto, está siempre a cero en nuestro ordendor.

- El *bit* 1 cuando contiene un 1 lleva al ordenador al modo de gráficos de alta resolución si los *bits* 3 y 4 del registro 1 contienen un cero.



***La conmutación de bancos es importante para poder aprovechar la memoria al máximo.***

**Registro 1:**

- El *bit* 0 de este registro puesto a cero indica al *VDP* que los *sprites* han de ser de tamaño normal. Un 1 en este *bit* hará que los *sprites* sean de tamaño doble, o sea, que cada *bit* que define al *sprite* dará valor a cuatro *pixels* de la pantalla.
- El *bit* 1 con valor cero indica que los *sprites* serán de 8x8 *pixels,* con valor 1 indicará que los *sprites* son de 16x16 *pixels.*
- El *bit* 2 no se usa en los *SVI-328* y *SVI-318* y está reservado para futuras ampliaciones.
- El *bit* 3 puesto a uno hará que la pantalla esté en el modo de gráficos de baja resolución *(SCREEN 2),* los *bits* que consiguen los otros modos de pantalla han de ser cero.
- El *bit* 4 conteniendo un uno consigue el modo de texto, si los otros *bits* de modo de pantalla están a cero.
- El *bit* 5 con un 1 habilita las interrupciones del *VDP.*
- El *bit* 6 habilita y deshabilita la pantalla, según contenga un 1 ó un cero.
- El *bit* 7 selecciona el tamaño de *VRAM.* Si es cero se dispondrá de 4 *Kbytes,* si es 1 serán 16 *Kbytes* de *VRAM.*

**Registro 2:**

Solamente se utilizan los cuatro primeros *bits,* del 0 al 3, con los que se tiene un número entre 0 y 15 que multiplicado por 400 hex da la dirección de la memoria *VRAM* donde comienza lo que llamaremos «tabla de denominaciones».

Si habéis seguido los artículos publicados en esta revista sobre la *VRAM* de los MSX y sus divisiones, podéis considerar esta tabla igual a las que en dichos artículos se mencionan como tablas 0, 5, 10 y 15.

Así tenemos que cada octeto de esta zona «denominará» que carácter es el que aparecerá en una determinada celdilla de ocho octetos de las que componen la pantalla. Todo esto lo veremos más detenidamente cuando veamos el contenido de cada registro en los diferentes modos de pantalla.

**Registro 3:**

Los ocho *bits* de este registro contendrán un número entre 0 y 255 que multiplicado por 40 hex nos dará la dirección de *VRAM* en que comenzará la que llamaremos «tabla de color» y que coincide con la tabla 11 de los artículos sobre la *VRAM* antes mencionados.

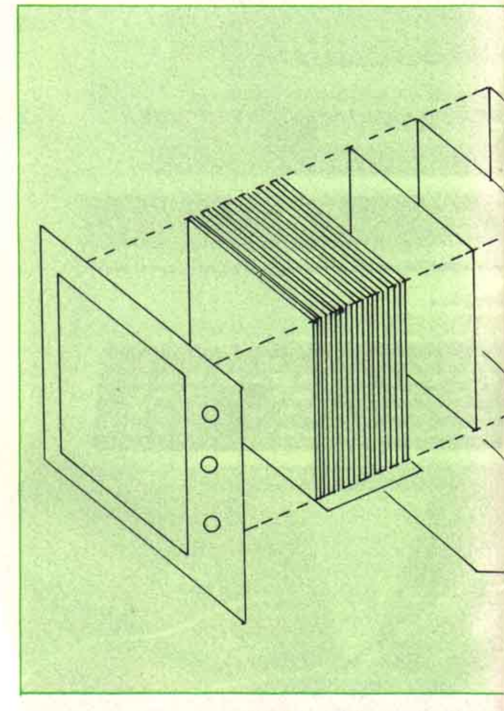


***Figura 1 Planos de la pan***

Cada octeto de esta tabla contendrá los colores de fondo y superficie de cada octeto de pantalla en el modo de gráficos de alta resolución como veremos al hablar de las tablas en este modo de pantalla.

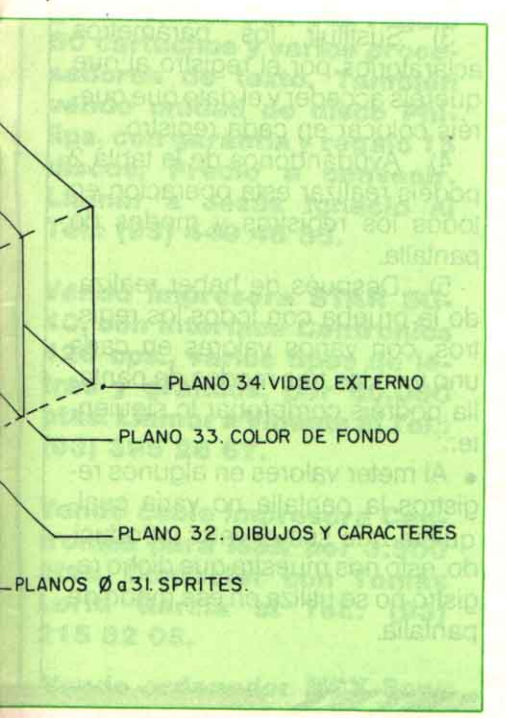
**Registro 4:**

De este registro se utilizan los *bits* 0 a 2, en los que hay un número entre 0 y 7 que multiplicado por 800 hex da la dirección de la tabla de «definición de patrones» que en los modos gráficos se transforma en un mapa de la pantalla y por lo tanto coincide con las tablas 2, 7, 12 y 17 de los ordendores MSX.

Más exactamente podríamos decir que los datos de esta tabla se verán reflejados en el antes mencionado plano 32 ó plano de patrones.

**Registro 5:**

En este registro se almacenará un número entre 0 y 127 que multiplicado por 80 hex nos dará la di-




*...alla con el TMS 9918 A.*

rección de la *VRAM* donde comenzará la tabla de atributos de *sprites.* Hay 4 *bytes* por cada plano existente para *sprites* donde se almacenarán las coordenadas de la posición del *sprite,* su color y el números de *bits* que lo componen.

**Registro 6:**

Sólo se utilizan los *bits* 0, 1 y 2 de este registro y contienen un número entre 0 y 7 que multiplicado por 800 hex da la dirección donde comienza la tabla de patrones de *sprites* que, como su nombre indica, contendrá las figuras que posteriormente podrán ser asignadas a un *sprite.*

**Registro 7:**

Los *bits* 0 a 3 de este registro forman un número entre 0 y 15 que corresponde al color del plano 33 que, como vimos antes, será el color de fondo en el modo texto o el color de borde en el modo de gráficos.

Los *bits* 4 a 7 contienen el color del texto en el modo de texto.

**Registro 8:**

Este es quizás el registro más importante del *VDP* ya que nos indica en cada momento el estado en que se encuentra el procesador de vídeo, por esto a este registro se le conoce con el nombre de «registro de estado» del *VDP.*

Mientras que los otros registros se utilizan para colocar en ellos valores adecuados a lo que queremos que haga el *VDP,* este registro se usará para leer en él datos que nos proporciona el *VDP.*

- El *bit* 7 es un banderín de interrupción que nos señala poniéndose a uno cuando el *VDP* completa un *scán* (un *scán* es una línea horizontal de *pixels*).
- El *bit* 6 es el banderín del quinto *sprite* que se alzará o pondrá a uno cuando haya cinco *sprites* en una misma horizontal.
- El *bit* 5 es el banderín de coincidencia de *sprites* y se pone a 1 cuando dos *sprites* coinciden o colisionan.
- Los *bits* 0 a 4 contienen el número de plano en que se encuentra el quinto *sprite* que coincide en una línea horizontal.

## Acceso al procesador de vídeo y a la VRAM

En el *BASIC* de nuestro ordenador hay dos instrucciones que nos permiten leer y escribir en la *VRAM.* Estas instrucciones son *VPEEK* y *VPOKE* que seguramente habréis utilizado más de una vez.

Por otra parte no teníamos una instrucción concreta que nos permitiera el acceso a los registros del *VDP,* cosa que sí ocurre en los ordenadores MSX con las instrucciones del tipo «*VDP(n)*» donde «*n*» es el número de registro al que queremos acceder.

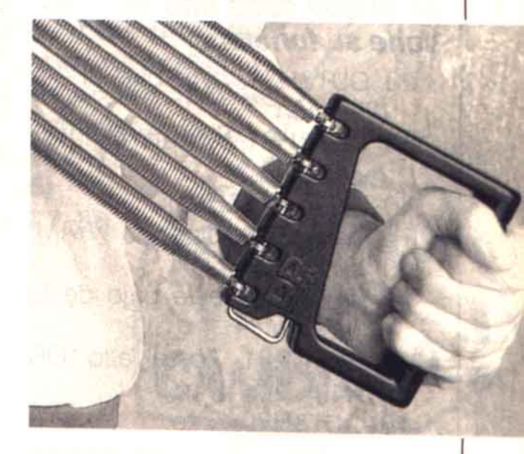
Pero hay una forma de acceder a estos registros que nos sirve también para acceder más directamente a la *VRAM.* Esta forma es a través de los *ports* que comunican en nuestro ordenador al *Z-80* con el vídeo *chip.*

Estos *ports* son los siguientes:

- *Port* de salida...80 hex...escribe datos en la *VRAM.*
- *Port* de salida...81 hex...por él se envía la dirección de *VRAM* o el número de registro a que se desea acceder y a veces también datos.
- *Port* de entrada..84 hex...lee datos de la *VRAM.*
- *Port* de entrada..85 hex...resetea el registro de estado para que la transferencia sea correcta.

Y la forma de usarlos:

- La primera operación antes de acceder al *VDP* mediante los *ports* ha de ser una lectura del *port* 85 hex para poner a cero el registro de estado y evitar así un acceso erróneo.
- Forma de leer en la *VRAM:*
  10 X=INP(&H85)



***Tanto el SVI-318 como el SVI-328 pueden llegar a tener 160K de RAM.***



***El manejo del VDP se realiza a nivel de bit, en el que cada uno tiene su función.***

20 OUT&H81, *byte* bajo de la dirección
30 OUT&H81, *byte* alto de la dirección
40 dato=INP(&H84)

- Forma de escribir en la *VRAM:*

10 X=INP(&H85)
20 OUT&H81, *byte* bajo de la dirección
30 OUT&H81, (*byte* alto OR &H40)
40 OUT&H80, dato

En la línea 30 se alza el sexto *bit* del *byte* alto para indicar al vídeo *chip* que se trata de una operación de escritura en la *VRAM.*

- Forma de leer el registro 8:

El registro 8 se lee cada vez que hacemos

X=INP(&H85)

esta instrucción coloca en la variable X el valor del registro 8 colocando en dicho registro un cero.

En los registros 0 a 7 del procesador de vídeo solamente podemos escribir, no obstante el valor de alguno de ellos lo podemos obtener de algunas variables del sistema como estas:

| Nombre de la variable | Dirección | Contenido |
|---|---|---|
| RG1SAV | FA07 hex | Registro 1 del VDP. |
| FORCLR | FA0A hex | Color del texto. |
| BAKCLR | FA0B hex | Color del fondo. |
| BORCLR | FA0C hex | Color del borde. |
| SCRMOD | FE3A hex | Modo de la pantalla. |
| SPRSIZ | FE3B hex | Tamaño del sprite. 0=8•8normal;1=8•8 aumentado 2=16•16 normal;3= 16•16aumentado |
| RG0SAV | FE3C hex | Registro 0 del VDP. |
| STATFL | FE3D hex | Registro 8 del VDP. |

***Figura 2***

- Forma de escribir en los registros:

10 X=INP(&H85)
20 OUT&H81,dato a escribir
30 OUT&H81,(registro OR&H80)

En la línea 30 se alza el *bit* 7 del número de registro para indicar al *VDP* que es un registro y no la *VRAM* lo que se desea acceder.

En el siguiente capítulo veremos lo que contiene cada registro en cada uno de los modos de pantalla y cómo se usan las antes mencionadas tablas en cada uno de los modos. Si no queréis esperar al mes que viene y averiguarlo por vuestra cuenta podéis seguir el siguiente procedimiento:

1) Escribir un pequeño programa que haga presentes en la pantalla la mayor cantidad posible de características de la misma en cada modo. (Ej. sprites, color, etc.).

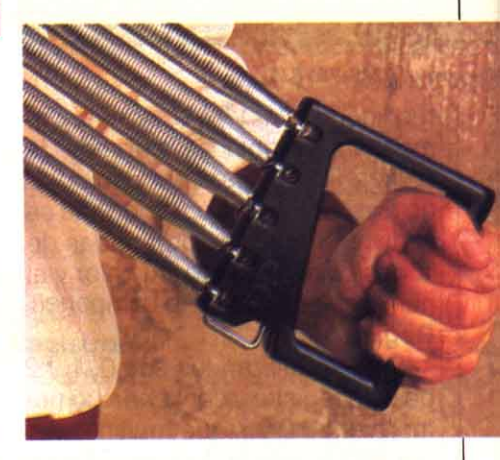
2) Incluir al final de vuestro programa las tres líneas que permiten escribir en un registro y que hemos visto antes.

3) Sustituir los parámetros aclaratorios por el registro al que queréis acceder y el dato que queréis colocar en cada registro.

4) Ayudándonos de la tabla 2 podéis realizar esta operación en todos los registros y modos de pantalla.

5) Después de haber realizado la prueba con todos los registros, con varios valores en cada uno y en los tres modos de pantalla podréis comprobar lo siguiente:

- Al meter valores en algunos registros la pantalla no varía cualquiera que sea el valor introducido, esto nos muestra que dicho registro no se utiliza en ese modo de pantalla.



***Solamente podemos escribir en los registros 0 a 7 del procesador.***

- Hay otros, en cambio, en los que cualquier valor introducido salvo uno produce cambios en la pantalla, ese valor que no produce un cambio en la pantalla es el que contiene el registro cuando habilitamos el modo de pantalla que estamos probando.

**Venerando Solís**

Vendo impresora Philips 80 columnas (fricción y tracción) en perfecto estado y garantizado. Con ella regalo 50 cartuchos y varios procesadores de texto. También vendo unidad de disco Philips, con garantía y regalo 15 discos. Precio a convenir. Llamar a Jesús Ignacio al Tel.: (93) 449 46 85.

Vendo impresora STAR SG-10, con interface Centronics 120 cps., varios tipos de letras y gráficos, por 60.000 ptas. Llamar a Vicente al Tel.: (93) 395 26 67.

Vendo cable impresora Centronics para MSX por 3.500 ptas. Contactar con Tomás Loriti García al Tel.: (93) 215 82 08.

Vendo ordenador MSX Sony HB-55P con ampliación de memoria de 64K y diversos juegos, todo en perfecto estado y 35.000 ptas. Llamar a Miguel al Tel.: (954) 61 26 36.

Intercambio o compro programas de Contabilidad, Facturación, Stocks para Spectravídeo con unidad de discos 605-A. Tengo DBASEII, SUPERBASIC, WORDSTAR, CTAS. CTES. BANCARIAS, etc. Escribir a Marco Lorente Duval. C/ Cuenca, 52. 46008 Valencia o llamar al Tel.: 325 89 30.

Vendo ordenador Sony HB-75P, como nuevo y económico por comprar otro MSX con disco. Llamar a partir de las 4 a Luis. Tel.: (91) 778 52 27.

Vendo equipo MSX, ordenador Sony HB-75P, unidad de disco Sony de 3.5 pulgadas, monitor ámbar Philips, impresora Philips y programas de Tratamiento de Textos y Base de Datos. Precio a convenir. Escribir a Manuel Farrer. C/ Gracián, 18. Calatayud (Zaragoza) o llamar al Tel.: (976) 88 10 91.

Vendo o cambio programas en cinta o cartucho. Escribir a Domingo Lamsfus Fernández. C/ Colonia San Fernando, 11, Motril (Granada).

Vendo Spectravideo SV-328. Incluyo cassette, mandos de juego, televisor portátil Philips en B/N, programas y libros, todo por 60.000 ptas. Llamar a Carlos al Tel.: (918) 22 88 96.

Por cambio de equipo, vendo Sony HB-75P en garantía y con muchos juegos en cinta y cartuchos, todo por 80.000 ptas. Escribir a Luis Sanz. C/ Latarsa, 22. 50006 Zaragoza.

Vendo ordenador Canon V-20 con poco uso y en perfecto estado. Regalo cables, manuales y muchos juegos. Todo por 50.000 ptas. Escribir a Elvis Martínez. Avda. Zamora, 99. Vigo, 11 o llamar al Tel.: (986) 41 45 25.

Vendo ampliación de memoria MSX de 16K casi nueva de Sony, por 5.000 ptas. Llamar a José al Tel.: (91) 255 05 56.

Vendo unidad de disco Philips VY 0010, impecable, con programas de tratamiento de textos, base de datos, control de stocks, presupuestos, pedidos, sistema DOS, ensamblador y juegos. Todo ello por 59.000 ptas. Interesados llamar a José Luis al Tel.: (93) 870 21 90.

Vendo libro «Juegos-Colores y gráficos para el ordenador TI-99-4A» todo en castellano por 2.500 ptas. Contiene el mapa de memoria. Escribir a José Arbona García. Pza. Juan XXIII, 1, Mislata (Valencia).

Deseo intercambiar programas para MSX. Llamar o escribir a Jesús López Alvarez. C/ Simancas, 2. Melilla. Tel.: (925) 68 77 38.

Intercambio programas MSX en cinta y en disco de 3.5 pulgadas. Escribir a Irene Juarros. C/ Garita, 19. 07015 Palma de Mallorca o llamar al Tel.: (971) 40 36 59.






# Complementación de VRAM

## Aplicación-Generación de vídeo inversor

Los registros dobles *HL* y *DE* se cargan con las direcciones final e inicial, respectivamente, entre las que se complementa la *VRAM*. Para la *SCREEN 0* la dirección final de la *VRAM* donde están definidos los caracteres es la *$1000* (4096 en decimal); la dirección inicial es la *$0800* (2048 en decimal). Colocando dichas direcciones en *HL* y *DE* y ejecutando el programa, se complementará toda la *VRAM* donde están definidos los caracteres. Al complementar todos también complementamos el carácter *BLANK*. Al estar la pantalla ocupada en las posiciones vacías (sin otro carácter) por el carácter *BLANK* se produce aparentemente el cambio del color de la pantalla y de caracteres. Con objeto de conseguir un efecto mejor comenzamos a complementar a partir de *$0918* (2328 en decimal) con objeto de no complementar el carácter *BLANK*, de esta forma no se

| | **Programa Fuente** | | Programa Fuente |
|---|---|---|---|
| | ORG | $F100 | origen para programa objeto |
| | LD | HL,$1000 | dirección última de caract. en VRAM |
| | LD | DE,$0918 | dirección inic. a modificar en VRAM |
| OTRO | CALL | $4A | rutina: lee dato de VRAM |
| | CPL | | complementamos el dato |
| | CALL | $4D | rutina: escribe el dato comp. en VRAM |
| | DEC | HL | aputamos a la siguiente posición |
| | LD | A,L | |
| | XOR | E | |
| | JR | NZ,OTRO | ¿es el último byte? |
| | LD | A,H | |
| | XOR | D | |
| | JR | NZ,OTRO | |
| | RET | | retorno |

modifica la parte de pantalla que no tiene caracteres.

Para la *SCREEN 1* cambia la localización de la tabla de caracteres; está entre las direcciones *$0800* y *$0000*. Para conseguir el mismo efecto que en la *SCREEN*, no modificamos la parte baja de la tabla donde está el carácter *BLANK*. Así cargaremos:

```
LD HL,$0800
LD DE,$0118
```

Estos valores pueden modificarse de forma que sólo complementemos los caracteres que de-

seemos (por ejemplo sólo los números).

Para los que no dispongan de un Ensamblador pueden cargar el progrma con éste en *BASIC:*

```
10 REM para CREEN 0
20 FOR I=0 TO 23
25 READ A
30 POKE(61696+I),A
40 NEXT I
50 DATA 33,0,16,17,24,9,205,74,
0,47,205,77,0,43,125,171,32,244,
124,170,32,240,201
```

Para la *SCREEN 1* el programa es idéntico variando sólo los primeros datos:

```
50 DATA 33,0,8,17,24,1,205,74,0,
47,205,77,0,43,125,171,32,244,
124,170,32,240,201
```

· Para modificar la dirección última que deseamos complementar tenemos que variar los datas segundo y tercero (L y H) y para modificar la dirección de comienzo de complementación, los datas cinco y seis (E y D).

## Utilización

Este pequeño programa queda colocado en la parte más alta del mapa de memoria libre. Debemos protegerlo con un *CLEAR.* Para poder utilizarlo en cualquier momento podemos definir su llamada con una *KEY.* A continuación recogemos un pequeño programa con todas estas sugerencias. El programa, una vez ejecutado, puede borrarse pues sólo inicializa unos punteros y define la *KEY 1.*

```
60 CLEAR 200,&HF100
70 DEF USR9=&HF100
80 KEY            1,"A=USR9(0)"+
   CHR$(&HD)
```

Después de ejecutar este programa tenemos en la *KEY 1* una nueva función. Al pulsarla se nos convierten los caracteres en negativo (vídeo inverso).

**Carlos Redondo Parral**
**León**

# Código Máquina

***Un nuevo capítulo de esta serie sobre el código máquina del Z80, esta vez dedicado únicamente a las instrucciones de rotación y desplazamiento. Se trata de unas instrucciones muy útiles, como veremos en capítulos sucesivos. Por ahora, baste decir que las operaciones matemáticas habituales (resta, multiplicación y división) se realizan utilizando la suma, la complementación y el desplazamiento.***

Antes que nada, hemos de haceros notar el sentido matemático de la operación de rotación o desplazamiento. Supongamos que estamos trabajando en el sistema numérico decimal, el habitual para todos nosotros, y consideramos el número 500. Si tenemos en cuenta que los ceros a la izquierda no cambian el valor del número, podemos suponer que trabajamos con un campo fijo de ocho dígitos, por lo cual escribiríamos 00000500. Como veis, simplemente hemos añadido cinco ceros a la izquierda, pero seguimos teniendo 500.

A continuación, definamos la rotación izquierda como una operación que toma el dígito de la izquierda y lo sitúa a la derecha. En este caso concreto, tomaríamos el cero de la izquierda y lo situaríamos a la derecha, obteniendo 00005000, esto es, 5000. Como veis, el resultado conseguido es multiplicar 500 por 10. En general, rotando a la izquierda en cualquier sistema numérico multiplicamos el número por la base del sistema. Así, en numeración binaria, que es lo que nos interesa, el resultado sería multiplicar por dos.

Veamos un ejemplo. El número binario 00001000 equivale al decimal 8. Si lo rotamos a la izquierda, obtenemos 00010000, lo que equivale a $16 = 8 \times 2$.

Si ahora consideramos la rotación a la derecha como tomar el dígito más a la derecha y situarlo a la izquierda, lo que conseguimos

es dividir el número por la base. Como ejemplo, tomad el dieciséis en binario que conseguimos antes, y rotarlo a la derecha (16 es 00010000: rotando nos queda 00001000 = 8).

Posiblemente alguno ya se habrá dado cuenta de que es muy fácil cometer un grave error. En el caso de rotación a la izquierda, si el dígito de la izquierda no es un cerc, al pasarlo a la derecha cometeremos un error. Y si se trata de rotación a la derecha y el dígito de la derecha no es un cero, también obtendremos un resultado erróneo (por ejemplo, si tenemos 5001 y lo rotamos a la derecha nos dará 1500, que no es lo mismo que dividir 0015, que no es lo mismo que 5001 multiplicado por 10).

Para poder controlar este error, CASI TODAS las operaciones de rotación y desplazamiento introducen el *bit* desplazado en el *flag* de acarreo, con lo cual, una vez realizada la rotación, basta con verificar este *flag* para saber si el resultado que tenemos es o no correcto. Además, el número almacenado en el *flag* de acarreo tras la operación será el resto de la división en el caso de la rotación a la derecha.

En algunas operaciones el *bit* desplazado se sitúa en su lugar de destino y en el acarreo, mientras que en otras se sitúa en el acarreo y el que había en el acarreo se sitúa en el destino de la rotación. Además, las instrucciones de desplazamiento no sitúan el *bit* desplazado en el otro extremo, sino que siempre va al acarreo, y el nuevo *bit* introducido no proviene del acarreo, sino que es un cero fijo o es el mismo *bit* que tenía antes. Lo veremos mejor con los ejemplos y con la tabla 1.

# código máquina

Vamos ya con el análisis general de las instrucciones:

*RLCA:* Las siglas corresponden a *Rotate Left Circular Acumulator* (Rotación Circular a la Izquierda del Acumulador). El *flag* N y el H se ponen a cero, y el acarreo se activa o no según cual fuera el *bit* siete antes de la operación. En la tabla 1 podéis ver qué es lo que hace: el







*bit* siete pasa a la derecha desplazándose los demás una posición a la izquierda. El *bit* siete se copia además en el *flag* C (acarreo). Es una instrucción de un solo byte.

*RLA: Rotate Left Acumulator* (Rotación Izquierda del Acumulador).

Los *flags* son afectados igual que en la anterior. En este caso, el *bit* siete pasa al acarreo, el acarreo para al *bit* cero y los demás desplazan una posición a la izquierda. Es también una operación de un *byte.*

*RRCA: Rotate Right Circular Acumulator* (Rotación Circular a la Derecha del Acumulador). Es la inversa de *RLCA.* El *bit* cero pasa a la posición siete y los demás se desplazan a la derecha. Además, el *bit* cero se copia en el acarreo. Los *flags* H y N quedan también a cero, y el acarreo varía según fuera el estado previo del *bit* cero. Es también una operación de un *byte.*

*RRA: Rotate Right Acumulator* (Rotación Derecha del Acumulador). Es la inversa de *RLA.* El *bit* cero pasa al acarreo, el acarreo al *bit* siete y los demás se desplazan un lugar a la derecha. Los *flags* quedan igual que en la anterior. De nuevo se trata de una operación de un solo *byte.*

*RLC s:* s representa a uno de los registros B, C, D, E, H, L, A y a la posición de memoria a la que apunta (HL), (IX+d) o (IY+d), donde d es el desplazamiento relativo respecto a la dirección a la que apunta IX o IY, como vimos en capítulos anteriores.

Esta operación afecta a todos los *flags:* N y H quedan a cero, el *flag* P/V indica sobrepasamiento, y los restantes *flags* (C, Z y S) quedan según sea el resultado de la operación. Como veis, una de las posibilidades es *RLC A,* cuya función es la misma que *RLCA,* pero con la diferencia del efecto en los *flags* y de que *RLC A* ocupa dos *bytes.*

En cuanto al desplazamiento de los *bits,* es el mismo que en *RLCA,* pero según la instrucción actúa sobre cualquiera de los registros señalados.

*RL s:* s es lo mismo que en la instrucción anterior, y el efecto producido en los *bits* es el mismo que en *RLA.* Los *flags* quedan igual que en la instrucción anterior.

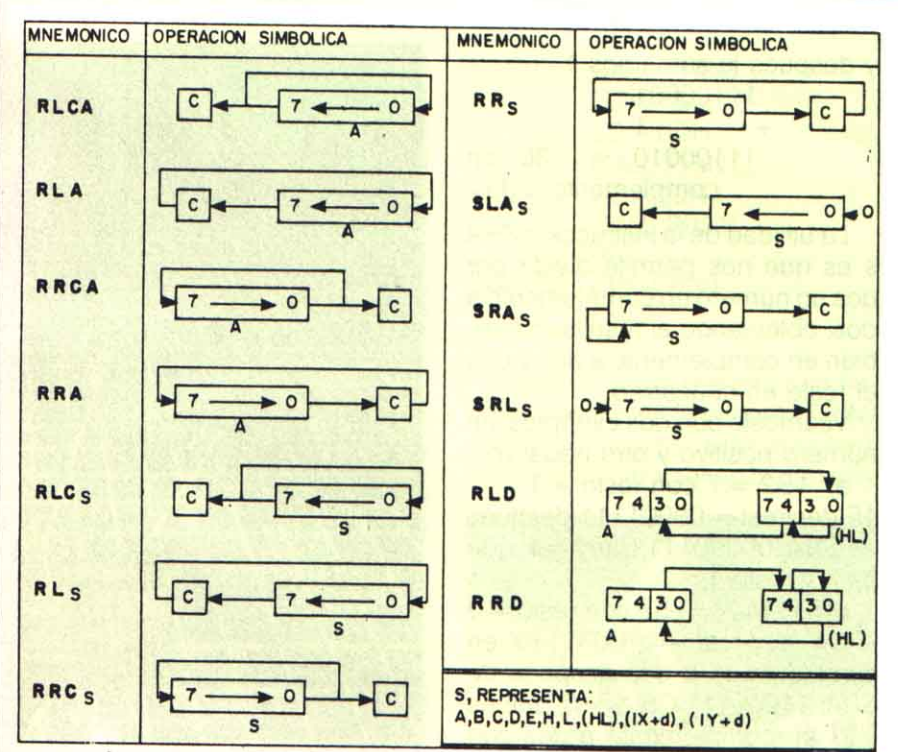
*RRC s:* s es lo mismo que en las instrucciones anteriores, y el efecto producido es el mismo que en *RRA.* Los *flags* quedan igual que en las instrucciones anteriores.

*RR s:* s es lo mismo que en las instrucciones anteriores, y el efecto producido en los *bits* es el mismo que en *RRA.* Los *flags* resultan afectados de igual modo que en las instrucciones anteriores.

*SLA s: Shift Left Arirmethic.* Desplazamiento Aritmético a la Iz-




quierda. La letra s representa lo mismo que en las instrucciones anteriores. En este caso, el *bit* siete se copia en el acarreo, los demás *bits* se desplazan a la izquierda y en el *bit* cero se introduce un cero.




Los *flags* quedan igual que en las instrucciones anteriores.

*SRA s: Shift Right Aritmethic.* Desplazamiento Aritmético a la Derecha. La letra s representa lo mismo que en las instrucciones anteriores. En esta operación, el *bit* cero de copia en el acarreo, los demás *bits* se desplazan a la derecha, y en el *bit* siete se realiza una copia del anterior *bit* siete, ahora *bit* seis. Esto permite dividir el número sin perder el signo, aunque esto lo comentaremos más adelante.

*SRL s: Shift Right Logical.* Desplazamiento Lógico a la Derecha. La letra s representa lo mismo que en las instrucciones anteriores, y se trata de la operación inversa de *SLA*. El *bit* cero se copia en el acarreo, los demás se desplazan a la derecha y el *bit* siete se pone a cero. Los *flags* quedan igual que en las operaciones anteriores.

*RLD: Rotate Left Decimal.* Rotación Decimal a la Izquierda. Esta operación permite trabajar con números codificados en *BCD*. Esta codificación consiste en representar un número decimal sin convertirlo a binario, de la siguiente forma:

| | |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

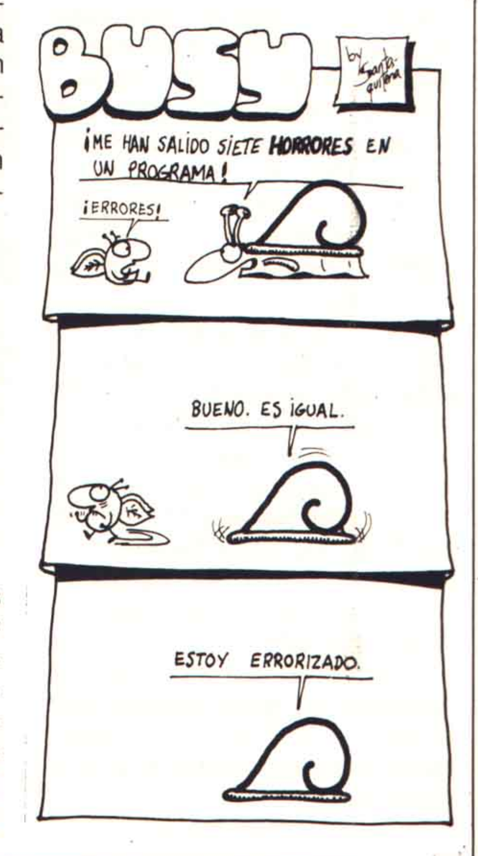
con lo cual los números binarios 1010, 1011, 1100, 1101, 1110 y 1111 no se utilizan. Como veis, para cada dígito decimal necesitamos cuatro *bits,* con lo cual podemos introducir en cada *byte* dos dígitos, uno en los cuatro *bits* de la izquierda, y otro en los cuatro *bits* de la derecha. Veamos un ejemplo:

El número hexadecimal 37H es igual a 16×3+7=55 decimal. Ahora bien, si consideramos que es un número codificado en *BCD,* para interpretarlo lo pasamos a binario y lo separamos en dos grupos de cuatro *bits:*

37H=00110111=0011 0111

Viendo ahora la tabla anterior, vemos que 0011 representa al 3 y 0111 al 7, luego 37H representa en *BCD* a 37 decimal.

Es muy importante que nos demos cuenta de que nuestro microprocesador *Z80* no distingue entre binario y *BCD.* Somos nosotros los que, al programar, debemos actuar según queramos trabajar con los números.



# código máquina

Volviendo a la instrucción que nos ocupa *(RLD)*, ésta no realiza una rotación de un *bit*, sino de grupos de cuatro *bits*, de forma que lo que conseguimos es desplazar un dígito *BCD* en cada rotación. Para ello se utiliza UNICAMENTE el acumulador y la dirección de memoria a la que apunta el registro *(HL)*, de esta forma:

1) Los *bits* 4 a 7 de *(HL)* pasan a las posiciones 0 a 3 del acumulador.

2) Los *bits* 0 a 3 de *(HL)* pasan a las posiciones 4 a 7 de *(HL)*.

3) Los *bits* 0 a 3 del acumulador pasan a las posiciones 0 a 3 de *(HL)*.

4) Los *bits* 4 a 7 del acumulador no cambian.

En la tabla 1 observaréis màs gráficamente esta operación.

*RRD: Rotate Right Decimal.* Rotación Decimal a la Derecha. Es la operación inversa de *RLD*. Tanto en este caso como en el anterior, resultan afectados todos los *flags* excepto el acarreo. Los *flags* N y H se quedan a cero, y el *flag* P/V indica paridad (P).

Una vez vistas las instrucciones, y antes de pasar al ejemplo, explicaremos más a fondo la instrucción *SRA s*.

Como ya comentamos en un capítulo anterior, una forma de representar los números es el complemento a dos. En esta forma, el *bit* siete indica el signo (0 = positivo, 1 = negativo), y los *bits* 0 a 6 el módulo del número en complemento a dos. El complemento a dos de un número se consigue invirtiendo cada *bit* y sumando uno. Veamos un ejemplo:

Para representar 30 decimal lo haríamos en binario normal, esto es 00011110. Si queremos representar −30, escribimos 30 y lo invertimos: 00011110

11100001

y después le sumamos 1:

```
  11100001
+        1
  11100010  =  −30 en
        complemento a dos.
```

La utilidad de la instrucción *SRA s* es que nos permite dividir por dos un número en complemento a dos, obteniendo el resultado también en complemento a dos y con el resto en el acarreo.

Veámoslo con dos ejemplos, un número positivo y otro negativo:

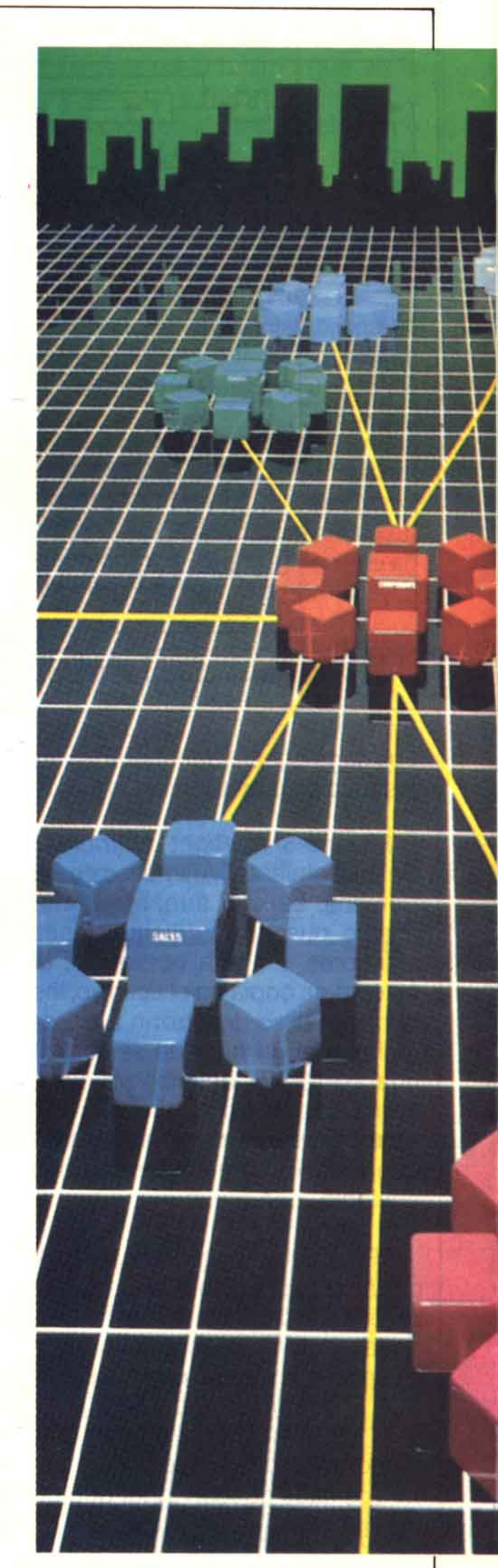
a) 15/2 = 7 con resto = 1
15 decimal = 00001111 después de *SRA:* 00000111 *Carry* = 1, que es 7 y resto 1.

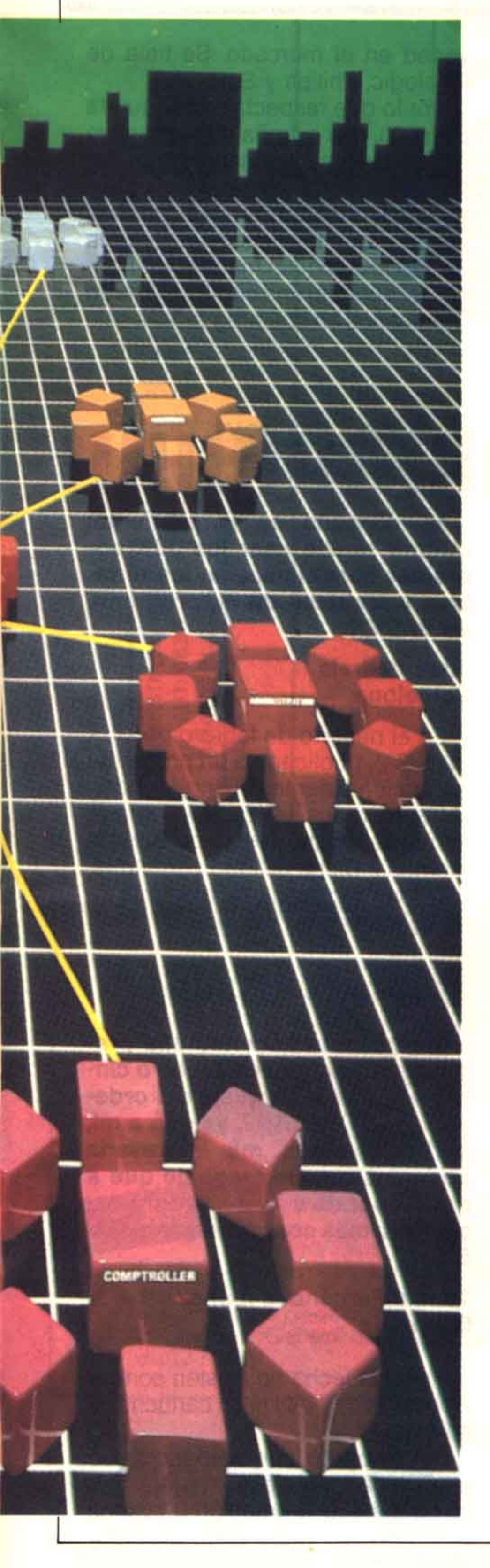
b) −114/2 = −57 con resto = 0
−114 decimal = 10001110 en complemento a dos después de *SRA:* 11000111 *Carry* = 0, que es −57 en complemento a dos con resto 0.

En cuanto a las dos últimas instrucciones explicadas *(RRD* y *RLD)*, queremos haceros notar que una rotación decimal equivale a multiplicar o dividir por diez el número en *BCD*.

Y ahora, el ejemplo. El programa que vamos a analizar tiene como finalidad inmediata imprimir en la pantalla una serie de ceros y unos que corresponden a los *bits* de una posición de memoria. Desde el *BASIC* podemos pokear el número que queramos analizar en la dirección adecuada, y el programa nos devolverá los 8 *bits* que lo forman (por supuesto, para números entre 0 y 255).

```
        ORG     B000H
PRINT:  EQU     00A2H
        ENT $
        LD HL,DIRECC
        LD B,08H
LAZO:   RL (HL)
        LD A,30H
        JR NC,SALTO
        ADD 01H
```





```
SALTO:  CALL PRINT
        DJNZ SALTO
        LD A,0DH
        CALL PRINT
        LD A,0aH
        CALL PRINT
        RET
DIRECC: DEFB 00H
        END
```

Analicemos: lo primero que hacemos es definir la etiqueta *PRINT* como la dirección de la rutina para imprimir un carácter en la pantalla.



Esta rutina necesita que le demos en el acumulador (AA) el código *ASCII* del carácter a imprimir, luego la misión de nuestra rutina ha de ser analizar cada *bit* del número en cuestión, y cargar en el acumulador el código *ASCII* del cero (30H) o del uno (31H) según corresponda.

*DIRECC* es la dirección de memoria en la cual pokearemos desde el *BASIC* el número a analizar. Así, al cargar esa dirección en *HL*, la instrucción *RL (HL)* rotará el número que queremos analizar.

Antes de empezar a rotar, cargamos el registro B con 8, ya que hemos de realizar la rotación 8 veces, una por cada *bit*. Utilizamos el registro B como contador de un bucle que cerramos con la ya conocida instrucción *DJNZ*.

Ahora cargamos el código *ASCII* del cero en el acumulador. Si el *bit* siete del número era un uno, este uno habrá pasado al acarreo, por lo cual no se realizará el salto en *JR NC*, y sumaremos al acumulador uno, obteniendo el código *ASCII* del uno (31H). En caso contrario, sí se realiza el salto, y el acumulador permanece con 30H.

Ya sólo falta llamar a la rutina de impresión de carácter y cerrar el bucle. Deetrás enviamos a la pantalla los caracteres de control 0DH y 0AH para simular la pulsación de la tecla *RETURN*. Y por supuesto, terminamos con *RET* para volver al *BASIC*.

Con la instrucción *DEFB* lo que hacemos es guardar una posición de memoria para el número a analizar. Aquí es donde deberemos pokearlo desde el *BASIC*.

El siguiente programa en *BASIC* carga el código máquina, os pide el número a analizar y ejecuta la rutina:

```
10 CLEAR 200,&HB000
20 S=0:FOR N=0 TO 29
30 READ D$:D=VAL("&H"+D$):
   S=S+D:POKE &HB000+N,D
40 NEXT N:IF S<>2544 THEN
   PRINT "ERROR EN LOS
   DATAS": END
50 DEF USR=&HB000
60 INPUT "NUMERO";NN
70 POKE &HB01D,NN
80 A=USR(0)
90 PRINT:GOTO 60
100 DATA 21,1D,B0,06,08,CB,16,
    3E
110 DATA 30,30,02,C6,01,CD,A2,
    00
120 DATA 10,F3,3E,0D,CD,A2,00,
    3E
130 DATA 0A,CD,A2,00,C9,00
```

# Rincón del lector

## CAMBIO DE ORDENADOR

**Tengo un ordenador de la primera generación y estoy pensando en cambiarlo por uno se la segunda con el fin de gestionar una pequeña empresa. ¿Sería suficiente un ordenador de este tipo para mis empeños? ¿Cuáles son los tres mejores de gestión en los campos de contabilidad, tratamiento de textos y bases de datos que entán o aparecerán en el mercado?**

**También me gustaría que me aconsejaran sobre el equipo que debe reunir y su coste aproximado.**

**Javier García Huesma**
**Valencia**

No cabe duda, que con el cambio saldría ganando bastante, al ser los ordenadores de la II generación algo más completos que los primeros. Sin embargo, hay que tener en cuenta que el software para la nueva generación apenas existe y, aunque sea compatible el ya existente con las nuevas versiones, faltaría probarlos a fondo para asegurarse de ello.

Los aparatos de la primera generación, están bastante bien para llevar una pequeña gestión, siempre y cuando tenga una unidad de disco. Este es, quizás el punto más importante, la unidad de disco.

Hoy por hoy, cualquier gestión por pequeña que sea requiere el uso de este importante periférico. Las razones son obvias, rapidez de acceso, facilidad de manejo, etc.

Referente al software, el tratamiento de textos y la base de datos más importantes que conocemos son el MS-TEXT, MS-BASE, IDEA-TEXT e IDEA-BASE, respectivamente. Los dos primeros son de Philips y vienen en disco y los dos siguientes son de Idealogic y vienen en cartucho. Con estos, le garantizamos que no tendrá que pensar en cambiarse de ordenador.

## IMPRIMIR COPIAS DE PANTALLAS

**Me gustaría que me informaran sobre la manera de hacer una copia de pantalla a impresora. Qué modificaciones habría que hacer, si ésta fuera el caso.**

**Patricia Torres**
**Madrid**

Desgraciadamente, los ordenadores MSX no tiene una instrucción que permita volcar la pantalla a la impresora. Sí se puede hacer con una rutina en código máquina, la cual publicaremos en meses sucesivos.

## COBOL PARA MSX Y UN LIBRO CON PROBLEMAS

**Os escribo para preguntaros si en el mercado existe algún tipo de programa que pueda trabajar en COBOL. Es caso de que existan, ¿dónde puedo conseguirlo y a qué precio?**

**Asimismo, quisiera plantearles un problema que me surgió al comprar un libro que vosotros comentásteis en una de vuestras revistas, concretamente «El Libro Gigante de los Juegos MSX» de Anaya Multimedia. El problema se presenta cuando intento ejecutar un programa del libro, que a pesar de haberlo repasado varias veces, no ejecuta. Os escribo a vosotros porque ya he mandado dos cartas a la editorial del libro y no me han solucionado el problema.**

**José A. Ortega Santana**
**Las Palmas de Gran Canarias**

Hasta la fecha no tenemos noticia de que existe un compilador COBOL para ordenadores MSX. Hoy por hoy, existen de Pascal (Sony), LOGO (Philips) y se presentará en breve un compilador de lenguaje C.

De cualquier manera, nada mejor que dirigirse a una de estas tres empresas, las cuales están en constante movimiento para conseguir la novedad en el mercado. Se trata de Idealogic, Philips y Sony.

Por lo que respecta a la segunda cuestión, nos extraña que en Anaya no te hayan solucionado el problema, ya que los libros recorren un largo camino de comprobaciones, repasos, etc. antes de ser puestos a la venta. De cualquier manera, mándanos (si puedes) una copia del listado o indícanos en qué programa te sale el error para intentar corregírtelo.

## USO DE LOS 32K DE MEMORIA

**Mi consulta es la siguiente: ¿Cómo puede usar los 32K sobrantes para almacenar las variables y matrices de mi programa en BASIC?**

**Juan Pedrals**
**Barcelona**

En el número de febrero de MSX (pág. 34), publicamos un programa excepcionalmente bueno, titulado «Más memoria utilizable» sobre la forma de utilizar los otros 32K de memoria. Dicha utilidad permite tener dos programas en memoria simultáneamente.

## COMPILADOR FORTRAN

**Me gustaría hacerles una pregunta, ¿hay algún cartucho o cinta que pase el lenguaje del ordenador a FORTRAN?, ya que a mí, particularmente, me interesaría poder utilizar un lenguaje que a nivel operativo y de programación sea más completo que el BASIC.**

**Tomás Fernández Ibáñez**
**Cantabria**

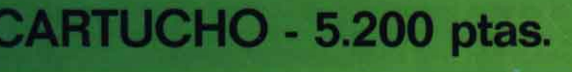
Hasta la fecha no existen compiladores FORTRAN ni en cartucho, ni en cinta.

Sin embargo, con la aparición de los ordenadores de al II generación, se barajan ya la creación de varios compiladores entre otros de FORTRAN.

Konami®

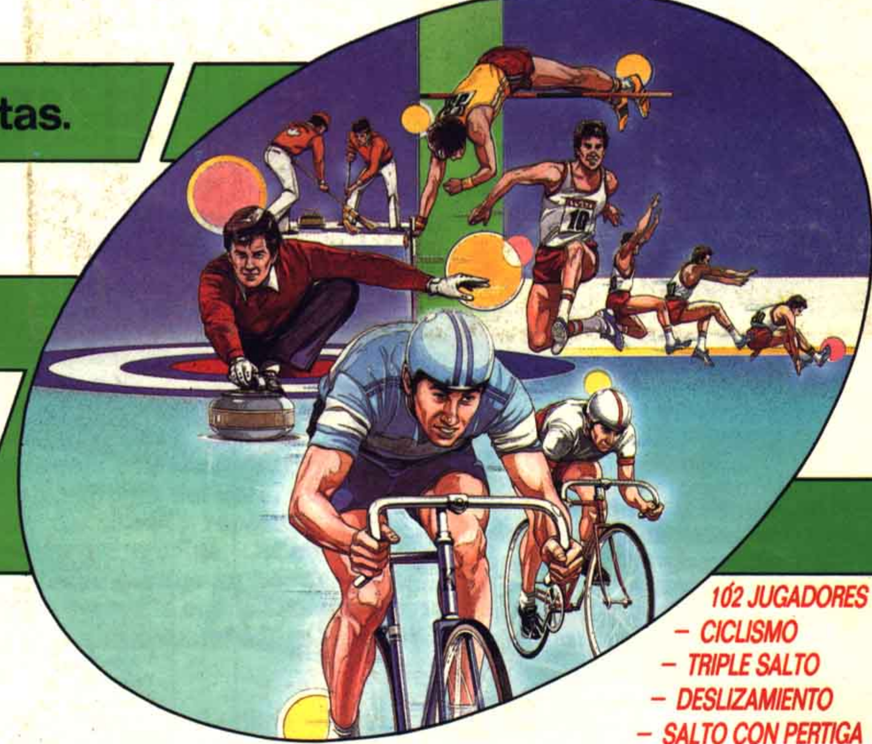
# Yie Ar KUNG~FU 2

*En el KUNG-FU 1, el maestro Lee aniquiló a toda la banda de Chop Suey. El malvado Yen Pey ha sido el único superviviente.*
*¡¡AQUI TIENES LA REVANCHA!!*









**1ó2 JUGADORES**
- CICLISMO
- TRIPLE SALTO
- DESLIZAMIENTO
- SALTO CON PERTIGA

SERMA

RECORTA Y ENVIA ESTE CUPON A: SERMA, C/. BRAVO MURILLO, N.º 377, 3.º A - 28020 MADRID - TELEFONOS: 733 73 11 - 733 74 64

CANTIDAD: ____________________
NOMBRE Y APELLIDOS: ____________________
DIRECCION: ____________________ CODIGO POSTAL: ____________________
POBLACION: ____________________ PROVINCIA: ____________________
FORMA DE PAGO: ____________________ ENVIO TALON BANCARIO ☐ CONTRA REEMBOLSO ☐